HyperCard, SuperCard, LinkWay, and Scanners

$4.95

HYPERLINK MAGAZINE

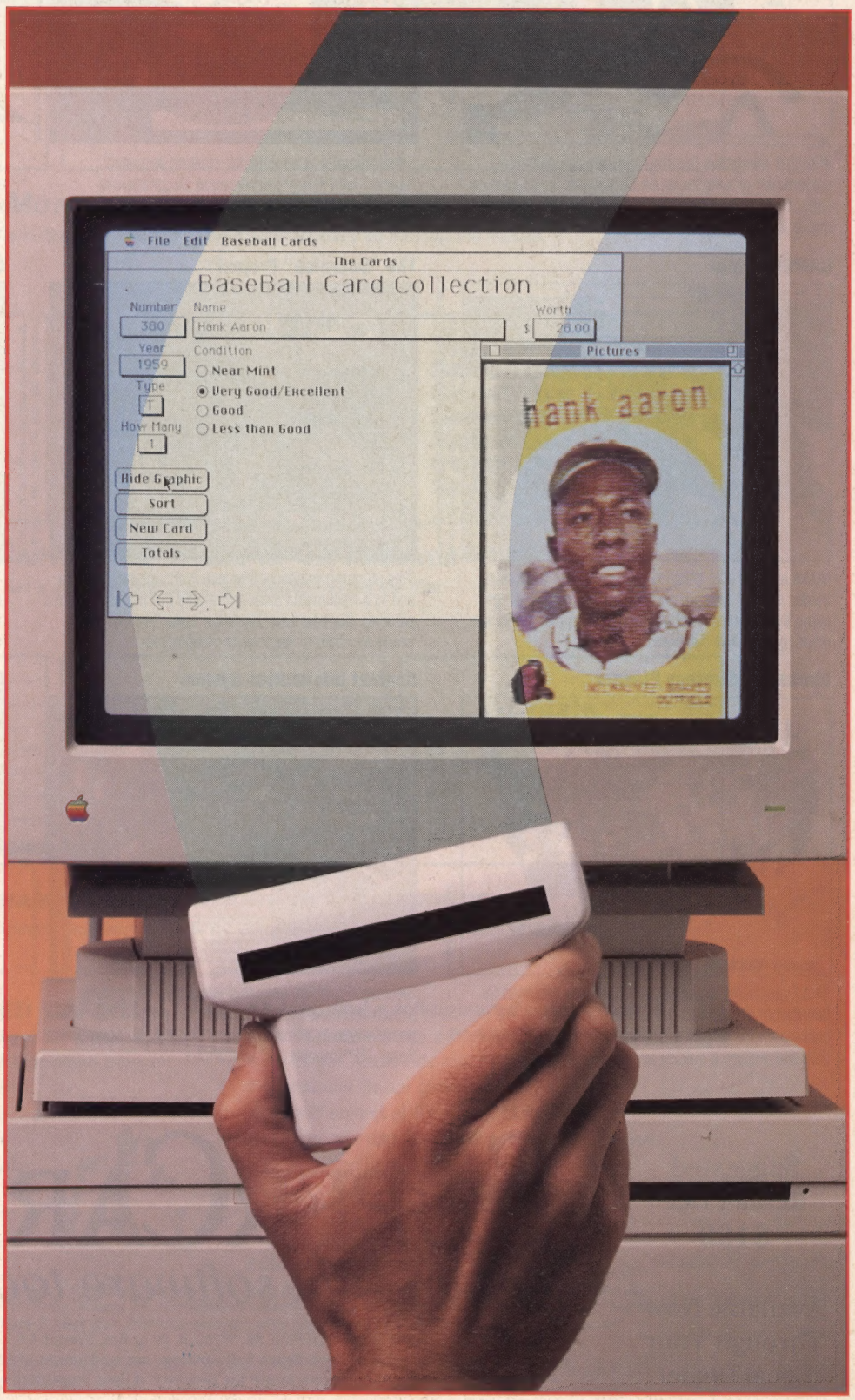
Sept/Oct 1989 • Vol. 2 No. 5

Focus on Hand-Held Scanners

Tracking Employees in *HyperCard*, *SuperCard*, and *IBM LinkWay*

Animation for All of Us

Plus Expanded Coverage



09
0 74470 74465 8

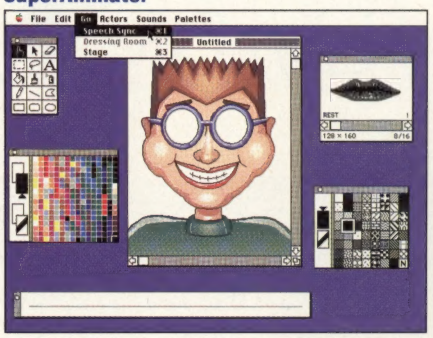
TRUE STORIES.

Oakley® Sunglass Selector

W.R. Robertson Advertising

With this interactive brochure, anyone can design a custom pair of sunglasses on screen and order automatically. This multimedia standalone uses color, digitized music and voice, animation and custom menus.

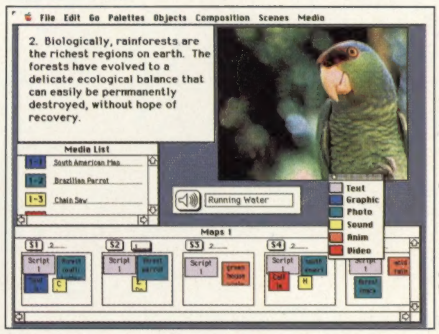
SuperAnimator™



Joseph Matthews, Bright Star Technology

SuperAnimator is a toolkit for creating animated "agents" with synchronized sound. A commercial product created in SuperCard, it incorporates the elements of a traditional Macintosh® application.

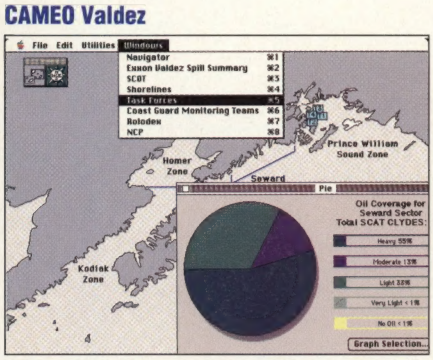
MediaWorks



Edward F. Boyle, Institute for Research on Learning

This multimedia composition environment was created as an Apple Classroom of Tomorrow project to allow 12-year-olds to create their own educational applications by combining information from a variety of sources.

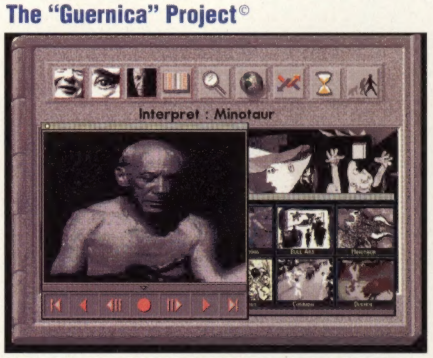
CAMEO Valdez



Concept and Graphics: National Oceanographic and Atmospheric Administration and Genesis Systems Programming: Peter C. Honebein

NOAA and the U.S. Coast Guard monitored the Exxon Valdez oil spill cleanup and conducted daily briefings using this interactive information system that combines maps, graphs and databases.

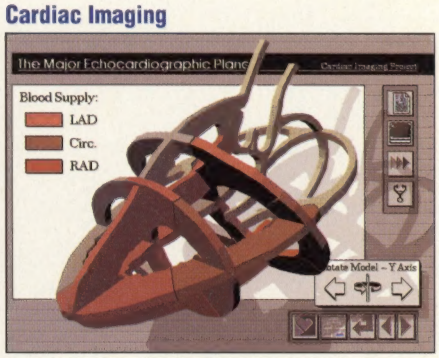
The "Guernica" Project©



The Guernica Project creative team: Robert Abel, Allen DeBevoise, Jonathan Gibson, Jerry Hesketh, Eric Martin, Morgan Newman

This stunning graphical interface is the interactive front end to a multimedia database related to Picasso's famous painting. Starting from *Guernica,* users can explore nearly any path of association imaginable.

Cardiac Imaging



C. Carl Jaffe, MD & Patrick J. Lynch, MS, Yale University School of Medicine

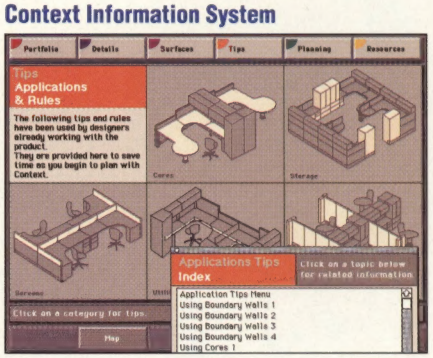
This interactive medical education project makes use of color graphics, interactive video, digitized sound and images created in Super 3D™ to teach residents and clinical staff at Yale University School of Medicine.

Marketing Strategy Planner

The HyperMedia Group

This hypermedia information system was designed for boardroom presentations and strategic analysis. By clicking on the graphics, speakers can pull up relevant data and analyze it quickly in response to questions.

Context Information System



Creative Interactive Media

Steelcase Furniture uses this interactive multimedia presentation and electronic catalog to present their new line of modular office furniture, "Context." It incorporates Super 3D™ models and animations.

Inigo At Home

Amanda Goodenough, AmandaStories™

Inigo achieved fame in the HyperCard® story stack entitled, "Inigo Gets Out." Author Amanda Goodenough easily converted her charming character to SuperCard and colored his world.

Suggested Retail Price **$199**

Available Now Through Your Local Dealer

SUPERCARD™

the personal software toolkit.

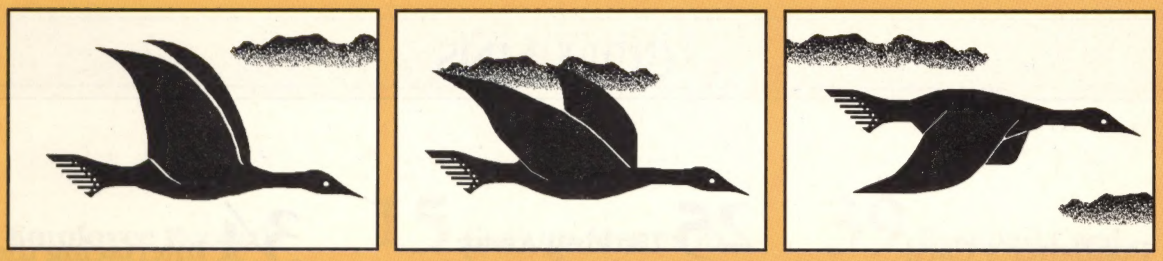
System requirements: Macintosh Plus, SE, SE/30, II, IIx or IIcx, System 6.0.2 or later. SuperCard, Super 3D and the Silicon Beach Software logo are trademarks of Silicon Beach Software, Inc. All other trademarks are the property of their respective owners.

SILICON BEACH SOFTWARE

P.O. Box 261430
San Diego, CA 92126
(619) 695-6956

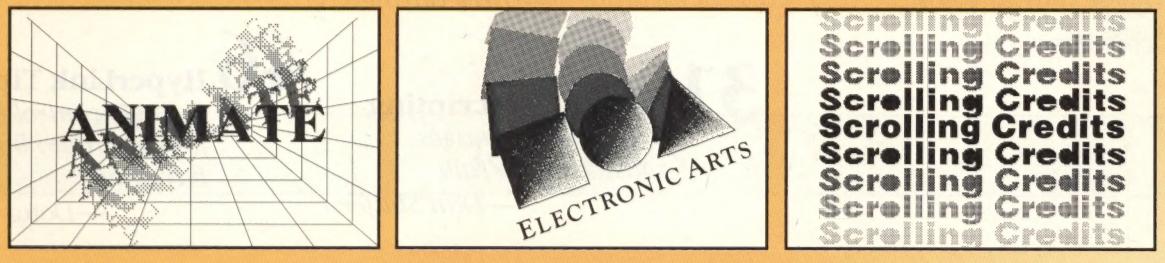
ANIMATION IN LESS THAN 1 MINUTE

WITH STUDIO/1™



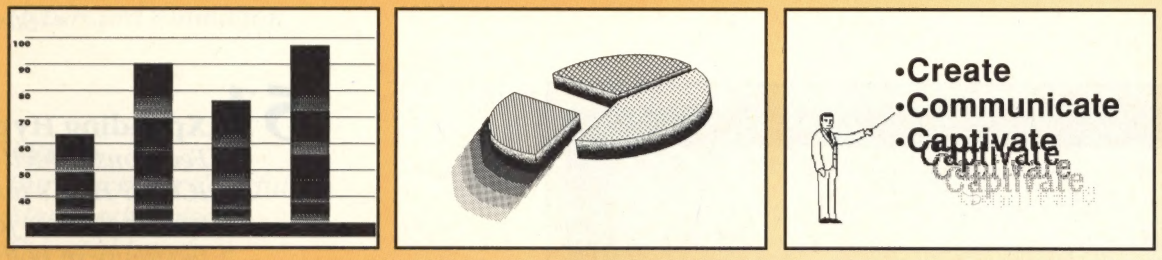
MOTION PICTURES IN 24 SECONDS

Automatic page flipping makes fluid animation as easy as moving the mouse.



CAPTIVATING TITLES IN 42 SECONDS

Set a start and end point, and Studio/1 does the rest.



DYNAMIC PRESENTATIONS IN 56 SECONDS

Studio/1 includes special animation effects for automatic transitions, fades and distortions.

1/2 PRICE LIMITED OFFER:

Get Studio/1 for only $75 (plus shipping and handling) when you trade up from any Macintosh paint or animation product. Offer good until Nov. 30, 1989. Demo disks also available for $10. Both offers good through Electronic Arts Direct Sales only. Call 800-245-4525 for details (M-F, 8-5 Pacific time).

All this, and a full-powered paint program too. *300 dpi editing, a PostScript® quality text layer, direct Apple Scanner® support, 3D perspective, editable Bezier curves, powerful selection tools and editing at 8 magnification levels, including zoomed-out mode. HyperCard® driver and animated slide show player included with Studio/1.*

ELECTRONIC ARTS®

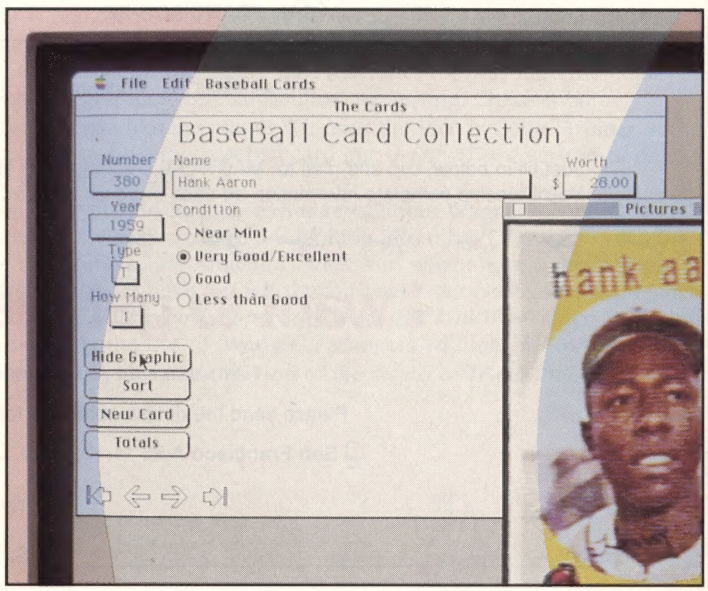
Studio/1 is a trademark of Electronic Arts. HyperCard, Macintosh, and Apple Scanner are registered trademarks of Apple Computer Inc. PostScript is a registered trademark of Adobe Systems, Inc. Requirements: Macintosh® Plus, SE, SE/30, II, IIx, IIcx; 2 disk drives or 1 disk drive plus a hard drive; System software 6.0.2 or later. Times represent animation renderings on a Macintosh SE. Other times may vary.

HYPERLINK MAGAZINE

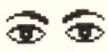
COLUMNS

FEATURES

REVIEWS



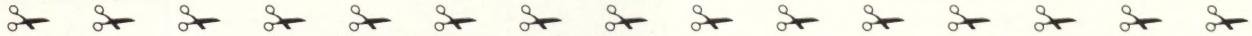
Picture yourself at MACWORLD Expo

There are  good reasons why you should attend the MACWORLD Expo nearest you. First, attending MACWORLD Expo means you don't have to spend lots of  and  trying to find the best solutions to your computing problems. Second, you can  and  the latest Macintosh products . . . demo hundreds of hardware, software and peripheral exhibits representing the future of  computing. Third, whether you use your  in the  , at  or at  you'll learn how to use it better. Fourth, if you develop, produce or sell  products, you'll have a unique opportunity to spot upcoming  . And fifth, you'll be able to  your skills, using one of the many   available, to practice what you've learned. So  the coupon below,  and  it for more information. And  this is the original Macintosh-exclusive show. Choose from these upcoming shows:

SAN FRANCISCO '90
April 11-13, 1990
Moscone Center
Brooks Hall/Civic Auditorium

BOSTON '90
August 9-11, 1990
Bayside Expo Center
World Trade Center

Just fill in below, clip and mail to: MACWORLD Expo, Mitch Hall Associates, 260 Milton St., Dedham, MA 02026.
You'll receive complete information on each show as soon as it's available. No obligation of course.



I want to get the big picture.

I am interested in: ❑ Attending ❑ Exhibiting

Please send me details about the following MACWORLD Expos:

❑ **San Francisco** April 11-13, 1990 ❑ **Boston** August 9-11, 1990

Name _____ Title _____

Company _____ Street _____

City/State/Zip _____ Phone _____

Sponsored by MACWORLD, the Macintosh ™ Magazine. An IDG Communications publication. MACWORLD Expo is an independent trade show not affiliated with Apple Computer, Inc. MAC, MACINTOSH and MACWORLD are trademarks of Apple Computer, Inc.

MACWORLD EXPOSITION

HYP

HYPERLINK MAGAZINE

Sept/Oct 1989 • Volume 2 No. 5

Publisher
David G. Brader

Editor
Roger Wood

Contributing Editors
Larry Burtness
Steve Michel
James Paul
Craig Ragland
Maurice Rizzuto
Dan Shafer

Director of Design
Marv Boggs

Director of R & D
William K. Balthrop

Controller
Mark Andersen

Public Relations
Ken Jackson

Advertising Sales
Carole Eversole
(503) 484-5157

HyperLink Magazine (ISSN 1045-4624) is published 6 times a year by Publishers Guild, Inc., P.O. Box 7723, Eugene, OR 97401. Telephone (503) 484-5157.

Editorial material should be submitted to HyperLink Magazine, P.O. Box 7723, Eugene, OR 97401. Submissions will not be returned.

Publisher's Card

Hyperbolic reflections on *HyperCard*. Is it true? Can you really run *HyperCard* stacks (via special software) on an IBM 80286-type computer under the Presentation Manager or Microsoft Windows?

Standing around in our MacWorld Expo Booth in Boston was pretty boring one afternoon until three guys showed up with a portable Compaq computer tucked under an arm saying they wanted to show me something interesting. "Yawn. OK, guys, what'cha got."

"We'll show you." With that, the Compaq portable computer was set up and with mouse in hand, click, click, and there on the LCD screen was the *HyperCard* "Home" card screen. They clicked on a button and up came the *HyperCard* graphics sampler stack. They activated the button tool and opened a script window.

My eyebrows went up several notches! I asked the most intelligent investigative reporting question I could muster in my excitement: "How did you do that?!" They filled my ears with "TechnoTalk" like: software pseudomachine, with C source complied to p-code, translated from the original stack on the Mac, etc.

"Can I get a copy to play with?"

"Not yet. If you are interested we will be back in touch." Poof, a cloud of dust, a blink of my eyes, and they were gone. Others saw this little demo at the show as well, so it wasn't just a poor tired publisher's fantasy.

This would be a great boon for Object Oriented Interface (OOI), Personal Software Development (PSD). Imagine building stacks that could run on Macs or IBM compatible machines! (Was that an earth tremor I just felt in the Bay Area?) If it is real, can it survive being Apple sauced (a new pseudo-legal term)—clever programmers like these will need deep pockets and clever attorneys as well. If this product proves to be more than vaporware, we will keep you informed.

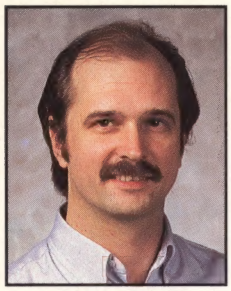
So what is real today? You will find coverage of Format Software's *PLUS* for the Macintosh (from Olduvai) starting in this issue. (Format Software is readying a *PLUS* version for the bigger IBM machines too.) I found the color and high resolution graphics capabilities most interesting. Check it out.

Hey, how do you like the cover? We used the ProViz Color Scanner with an 8mm camcorder to scan the baseball card into a *SuperCard* version of the baseball card stack in this issue.

Next issue we plan to include coverage of Roger Wagner's *HyperStudio* that runs on the Apple IIGS, and we are looking into other OOI PSD stuff (like *UltraCard* for the Commodore Amiga) as well. And what about a new version of *HyperCard?* We should have the word two issues from now. Meanwhile, this issue is chock full of *HyperCard*, *SuperCard*, and *IBM LinkWay* things to do and learn about, including the use of hand-held scanners with *StackProjects*.

I hope you enjoy this latest edition of the world of *HyperLink*.

David G. Brader



Roger Wood
Editor

Finding an Easier to Use Find

I have a *HyperCard/SuperCard* programming tip, but it doesn't really fit into the "Fields of Intelligence" or "Buttons & Bows" categories. It's just a general tip. Thought you might like to publish it, though.

When designing a stack or *SuperCard* application that will be used by HyperTalk illiterate users, it is important for the user interface to be intuitive. If you want your user to be able to find data in a stack, there are three ways: bad, better and best.

Bad—

```
on mouseUp
  domenu "find"
end mouseUp
```

This option is a bad one because it results in a message box with `Find""` and a flashing cursor between the quotes. While *HyperLink* readers will immediately know what to do here, the novice may be confused. They may not know to type in their search word followed by Return.

Better—

```
on mouseUp
  ask "Find what? with ¬
  "Type search word here"
  if it is empty
  then exit mouseUp
  find it
  if the result is ¬
   "not found" answer ¬
   "Couldn't find match"¬
   with "OK"
end mouseUp
```

This option is better because the Ask dialog box is handled much more intuitively by the user. It is somewhat weak, because once the `find` is executed, the user can't simply hit the Return key to find the next occurrence of the word. The ideal user interface is accomplished with the Ask dialog box, but with the ability to type the Return key to find multiple occurrences of the text being searched. This can be accomplished by putting the `find` command into the Message box, so that when the Return key is hit, the `find` command in the Message box is executed. Unfortunately, when you put something into the Message box, it appears. The trick to get around this is to move the Message box off screen, then put the `find` command into it, then hide it, then return it to its original coordinates:

Best—

```
on mouse Up
  ask "Find what?" ¬
  with ¬
  "Type search word here"
  if it is empty
  then exit mouseUp
  put it into ¬
  searchstring
  find string ¬
  searchstring
  if the result is ¬
  "not found"
  then answer ¬
  "Couldn't find match" ¬
```

```
  with "OK"
  put the loc of ¬
  message into ¬
  originalLocation
  set the loc of ¬
  message to 0,-300
  put "find string" && ¬
  quote & searchstring ¬
  & quote into ¬
  message box
  hide message box
  set the loc of ¬
  message to ¬
  originalLocation
end mouseUp
```

Now, when the user wants to find something, the Ask box comes up, they type in their search string, the `find` is executed, and subsequent returns `find` the next occurrence of the text!

Hope this helps!

David Altfeder
Carolina Beach, NC 28428

All in all, a very good solution, David. We do have one quarrel with using the Ask box and that is that it limits experienced Mac users who want to use the standard clipboard to paste in the search string. The fact that pressing ⌘-V puts a lower case "v" into the Ask dialog, instead of pasting what is on the clipboard has long been a pet-peeve of ours. Hopefully, this will be remedied in a future release of HyperCard. *Of course, an experienced user could always use the Message box, but why not have the best of both worlds?*

Easy Lock to Pick

Last year I bought a copy of the first issue of your magazine (Apr/May 1988). I had just purchased a Macintosh Plus system a few months before (my first exposure to computers). I read your magazine with interest, although I must admit that I did not comprehend much of what it was all about at that time. Last week, I picked it up again because I saw that it had an article on copying art from Thunderscan into *HyperCard* (something I was attempting to do at that very time). While browsing through the magazine again, I found the article by Joan Donaldson called "Short Stacks." Just for fun, I tried to set up a diary according to the article with a "Lock" button requiring a password. It worked great except that there are simple ways to bypass the lock. By simply choosing "Next" from the Go menu, my diary is opened to curious eyes (not that I have much to hide). Also, anyone who has become familiar with creating buttons would have no trouble opening the script and finding the password. Is there some way around these problems to make my *HyperCard* stacks more secure, or will I just have to assume that my two teenage daughters will never discover the not-so-hidden secrets of *HyperCard?* (Ha!) Regardless, the article opened up another whole world to me that I had not yet discovered: scripting. Thanks for the information!

Don Richter
Petersburg, AK

You're welcome, Don. Yes, the password protection in that stack leaves something to be desired. It wasn't intended to be a fool-proof lock anyway, just an idea for people just getting started with HyperCard. *The lock could be made to function better by placing a* `doMenu` *handler in the stack script that reads:*

```
on doMenu which
  if which = "Next" or ¬
  which = "Prev" or ¬
  which = "Last"
  then beep
  else pass doMenu
end doMenu
```

There are many ways this could be circumvented as well, such as using the arrow keys. A similar sort of `on arrowKey` *handler could be devised to handle that case, but as you point out, as long as the script is available the password can be learned. You could make use of* HyperCard's *built-in protection (the "Protect Stack..." item under the File menu). Of course, there is an XCMD called "Deprotect" on many bulletin boards that renders this protection useless, so it is of limited value as well.*

In other words, a fool-proof password is very difficult to devise, but you might be able to turn it into an interesting puzzle both for yourself and your teenage daughters.

Just in Time!

July/August issue—Wow! Just in time! I am finding at least one article in each issue, of strong interest. How did you know I was just starting an interactive videodisc project?

Robert Fuller
Lincoln, NE 68502

Glad we were timely, Robert. There are a lot of people out there trying to put together interactive videodisc projects—frankly we've been pleasantly surprised at the amount of interest this particular article has generated. Do let us know what you want and we'll try to keep the information coming your way.

Cover *UltraCard?*

A recent development in the Amiga software market prompted me to pick up a copy of *HyperLink.* This was the July/August 1989 issue, and I was hoping for at least some useful information and/or a hyper-environment overview. You see, neither do I own *HyperCard* or *LinkWay,* not, for that matter, *SuperCard.* I am, however, using my Amiga mostly for productivity applications and was recently surprised to find a very interesting looking *HyperCard*-like program on my dealer's shelves, called *UltraCard.* I am now trying to find out how hyper-language compatible its script language is, and if a magazine such as yours might possibly begin covering it for Amiga owners. In case you are wondering what makes me think you might do just that, it's something said in that issue's "Publisher's Card" to the tune that the editorial realm of *HyperLink* being all *HyperCard*-like stuff.

On the other hand, Macintosh and IBM type magazines have never deigned to mention the Amiga without a great deal of sneering, and you might be just like all the rest of them. Even though over one million Amigas have sold so far, even though the Amiga has an excellent REXX language port available for it which links many different applications, sneering seems to be an important self-defense reflex for Macintosh and IBM computer owners and the magazines that service them.

But then, maybe, you will be different. I sure hope so.

Christopher Herd
Colorado Springs, CO

If you read Dave Brader's "Publisher's Card" in this issue you will see that we are indeed very interested in covering this new piece of software. The Amiga has been a leading-edge machine, both by providing multitasking when the machine was released, and in areas of video control, music, graphics, and a number of other applications. Rest assured that we will provide our readers with the latest on this ever evolving genre of Personal Development Software on many hardware platforms.

More Rave Reviews for the #1 SQL Database.

PC WEEK

Oracle for Mac Solves DBMS Puzzle

SQL/HyperCard Combo Produces Sophisticated Applications

By Susan Janus

Database giant Oracle Corp., of Belmont, Calif., recently rounded out the list of platforms supported by the company's relational database-management system (DBMS) with the release of Oracle for Macintosh.

The new Mac DBMS teams the power of Oracle's Structured Query Language (SQL) with HyperCard's customizable and easy-to-use interface.

This combination allows Mac developers to create sophisticated applications that appear simple to users—a unique software-development opportunity unavailable on any other platform, including the PC, according to early corporate users we contacted.

In addition, users said, the network version of Oracle for the Mac allows the advantages of the HyperCard interface to serve as a front end to existing corporate relational databases residing on other platforms, such as mainframes and minicomputers.

Consequently, the Mac software has great potential for [illegible]

Oracle for Macintosh

DBMS Brings New Power and Face to Mac Application Development

WHAT CORPORATE BUYERS LIKE

- Offers SQL capabilities
- Uses HyperCard to mask complexity of database applications
- Serves as front end to relational databases on other platforms
- Can link different Oracle database platforms with network version
- Provides good documentation and support

oping database-manageme... systems to connect departmental Macs to larger systems. Jeff Menz, a systems analyst for a scientific R and D organization, is currently using Oracle for Macintosh to develop a system that will give the company's numerous Mac users access to an Oracle database running on a VAX. Using the networking version, Menz will link departmental AppleTalk networks to the VAX over a TCP/IP network.

Oracle Provides the Groundwork

The key benefit of Oracle's product is that "all the fundamental technical work—the protocols, the compatibility with the AppleTalk network—has been done by Oracle," Menz said. "I just have to build the HyperCard interface and do the database design work."

The result will be an Oracle relational-database application on the VAX with a HyperCard front end that Mac users feel comfortable with, he explained. Also, the company will have one logical relational database instead of numerous fragmented local databases. This allows tighter security, improved data integrity and better performance, he added.

"ORACLE for Macintosh is a well-designed product ... a programmer can, with just an hour's training, create a database on a host with a simple Macintosh interface in three minutes—honest." ★★★★

MacUser, June 1989

"ORACLE for Macintosh is exciting for companies that have Oracle (or DB2) databases on a mainframe and want a practical Mac data entry and development system."

Mac Week, March 14, 1989

"Get yourself a copy of ORACLE for Macintosh and get a flexible toolkit that can be molded to solve just about any database problem."

MacGuide Magazine, June 1989

ORACLE®

COMPATIBILITY • PORTABILITY • CONNECTABILITY

Call 1-800-ORACLE1, ext. 9269 today.

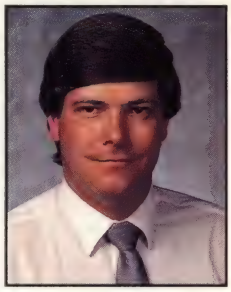
Until 10/31/89, for $299[1] you can access the number one SQL database from either HyperCard or 4th DIMENSION. With our 30-day money-back guarantee, the only thing you stand to lose is a great buy. Call today.

Dan Shafer, *Author of Hypertalk Programming*

Guy Kawasaki, *President of ACIUS, developers of 4th DIMENSION*

[1]Stand alone version licensed for developers only. Requires Macintosh SE or II with 2 Megabytes of RAM, 5 Megabytes hard disk space, floppy disk drive, and HyperCard 1.2. Includes 30-day installation support, ORACLE database, Hyper*SQL (HyperCard Interface), SQL*Plus, OCI and Pro*C (Macintosh Programmer's Workshop is required for programming usage), System Stacks and Example Stacks. Full networking version is $999 and includes SQL*Net (for database communications), Async, 3270, DECNet, TCP/IP support, Alisa System's T 5 Snet DECnet protocol and drivers and Kinetic's TCPort TCP/IP protocol and driver. Accessing database software on other machines requires a separate protocol handler and gateway software for the other machine. Call for additional information.

An Application in HyperCard, SuperCard, *and* IBM LinkWay *for Organizing...*



Employee Records

by William K. Balthrop

One thing common to large and small companies alike is keeping records on employees. Even if you employ only a few people, having a standard way to organize personal and business related information about each employee can save you a lot of headaches.

You have a computer. Let it do all the hard stuff, such as trying to remember when John Smith got that raise last year, or whether that ex-employee trying to get re-hired was an asset. Burn all those file cabinets full of paper, and get with the program—*HyperCard*, *SuperCard*, or *IBM LinkWay* that is.

We've created three versions of an "Employee Records" application to demonstrate how to approach computerizing this record keeping. Each version takes advantage of each software system's particular strengths, and still maintains similar functional operation between the versions. Because of the size and complexity of these applications, the complete object information and scripts will not be listed. Instead, we will include a detailed description of the applications and focus on two of the more interesting aspects—sorting and scanning. We'll explain how each version accomplishes these tasks to aid you in designing your own applications. You may receive any of the three versions of the complete applications on the *StackProjects* disk. [See the inside back cover for ordering information—Ed.]

Information—What to Track?

We have divided the information to be recorded for each employee into into four categories: Address, General, Financial, and Benefits. This system is designed so that all of the information for each category may be entered on a single screen, without any screen becoming too cluttered.

William "Kelly" Balthrop is the Director of R & D at HyperLink *and has been developing and publishing computer software for the last decade. Kelly serves as a consultant for the Object Oriented Interface (OOI) genre of software* (HyperCard, SuperCard, LinkWay, *etc.).*

Address

The Address screen contains all of the important elements which describe the employee's location—both home address and the division and department in which the employee works. The supervisor's name, along with the employee's work phone and extension are also included (see Figures 1, 2, and 3). A "Comments" field at the bottom of the screen allows for the entry of any

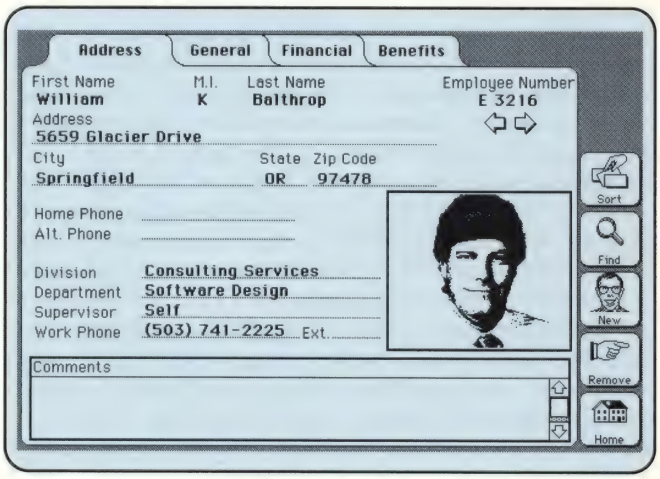


Figure 1

Here is the HyperCard *version's "Address" Card.*

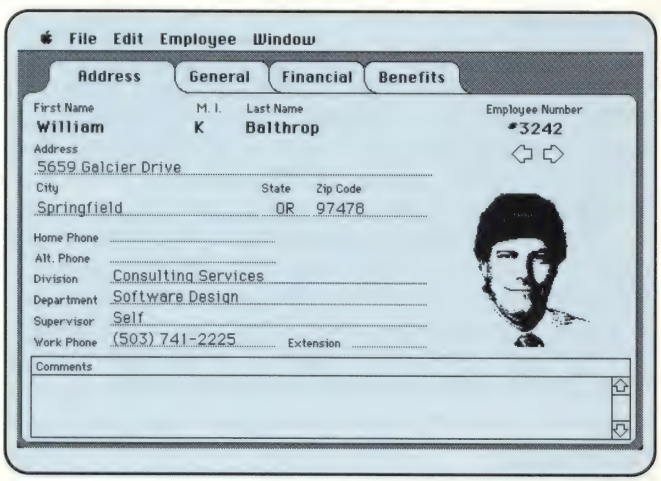


Figure 2

The SuperCard *version uses an Employee menu instead of buttons to allow the user to access to tasks such as sorting.*

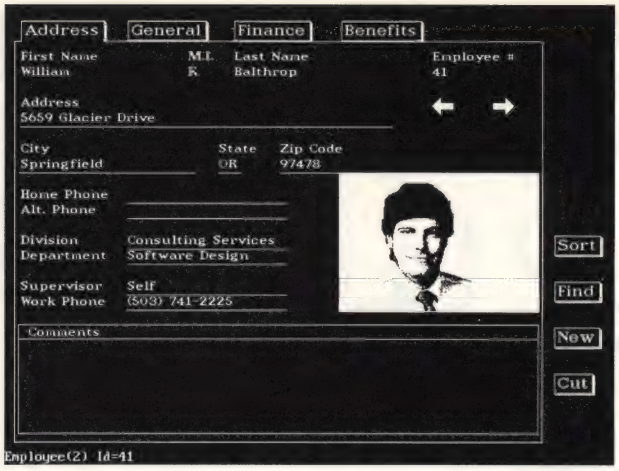


Figure 3

The IBM LinkWay *version of "Employee Records" includes all of the functions and operates much like the* HyperCard *version.*

information that has not been provided for in fields, but may be of importance to your company.

Unique to each employee is an employee number. This number is generated by the computer (it is the card or page id number of the Address screen), and is never changed once assigned. The employee number is used for navigating between the categories of information and is displayed in the upper-right corner of each screen. Below the number are two arrow buttons. Clicking on these buttons takes you to the next or previous employee record. When navigating from one employee record to another you remain in the same category. To change categories you must select one of the four options at the top of the page. Along the top of each page are four buttons labeled with the four categories. Simply click on any of the buttons to see that category of information for an employee.

Notice also in Figures 1, 2, and 3 a box on the right side of the screen with a picture in it. This allows you to add a picture of our employees. The use of this area is optional because it requires access to a scanner. We'll discuss the various methods of scanning supported by these applications later in this article.

General

The General information screen, Figures 4, 5, and 6 contains information which is important, but doesn't fit into the other three categories. This includes the names of any spouse and children. The employee's date of birth, date hired, date terminated, sex, and Social Security number are also included.

You will want to be able to recall whether you would consider that person for rehire. You can do so by setting the "Recommended for Rehire" check box on the right side of the screen. A large comments field allows for the entry of notes, or information about an employee which has not been provided for on this screen.

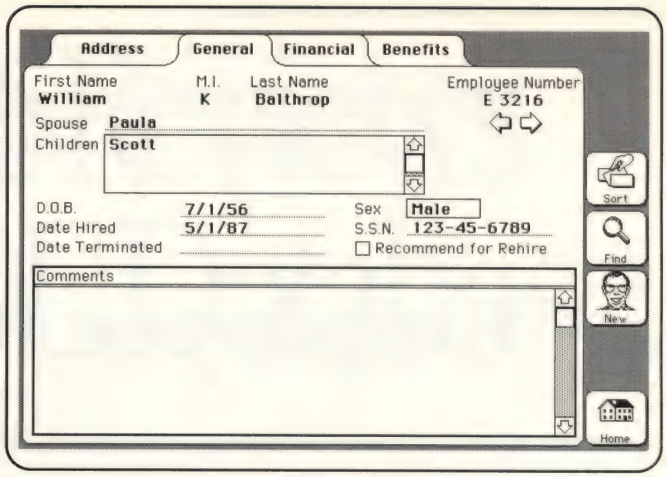


Figure 4

This HyperCard *screen lets you keep track of general information. The "Remove" button is not accessible here—all records deletions must be initiated from the "Address" Card.*

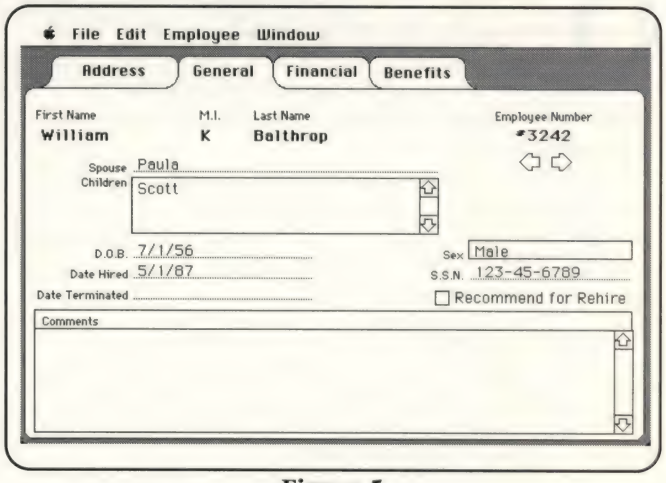


Figure 5

This "General" screen is from the SuperCard *version.*

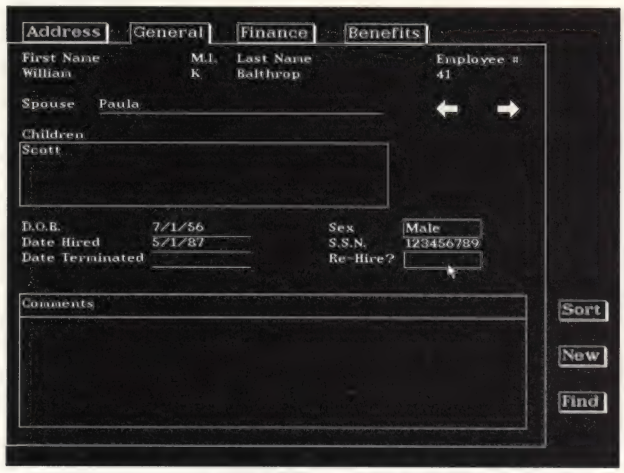


Figure 6

The LinkWay *version's "General" page is virtually identical to the* HyperCard *version.*

Financial

The Financial screen (Figures 7, 8, and 9) is intended to keep a general history of the employee's financial status with the company. The financial information comprises marital status, the number of exemptions (from a W-4 form), the current position, the date the employee started at that position, the current wage or salary, and the period upon which the wage or salary is

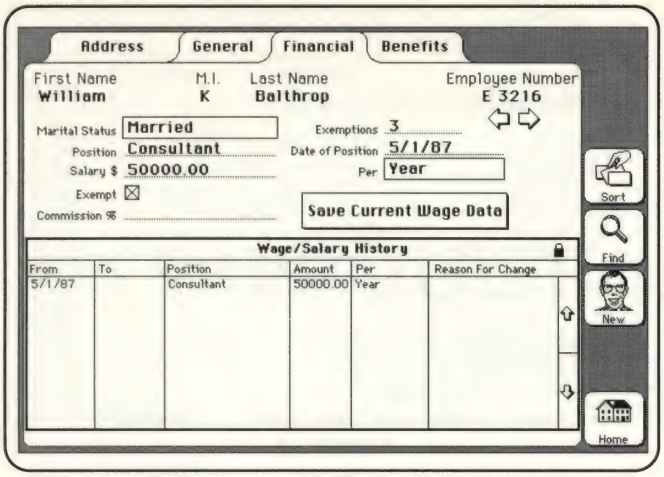


Figure 7

Clicking the "Save Current Wage Data" button in the HyperCard *version's "Financial" screen automatically enters the current data into the "Wage/Salary History" field.*

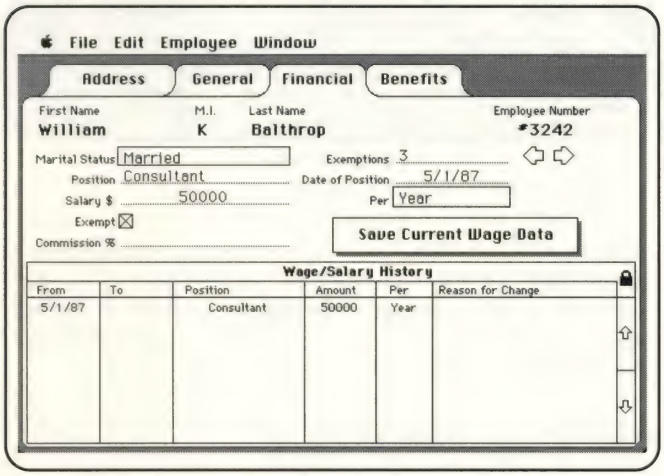


Figure 8

The SuperCard *version is virtually identical to* HyperCard.

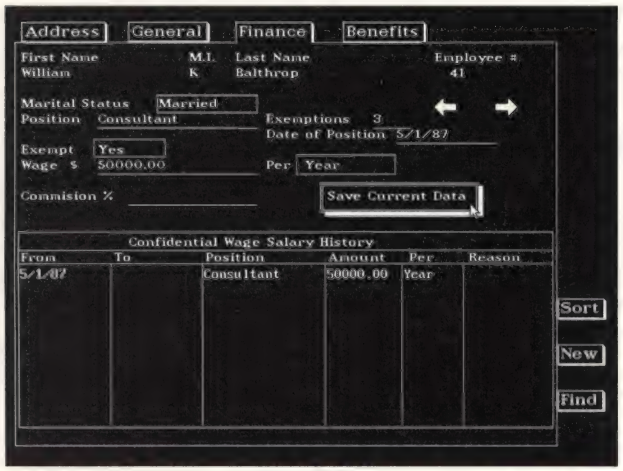


Figure 9

This is the LinkWay *version's "Financial" screen.*

based, and the percent commission (if any) the employee earns.

Between the "Wage" field and the "Commission" field is a button called "Exempt." Employees paid a salary as opposed to an hourly wage are considered exempt, and this check box is selected for employees who earn a monthly or annual salary.

When the button is selected the heading for the "Wage" field is changed to "Salary," the salary period is set to "Year," and this "Salary" field is cleared. Enter the annual salary into this field. If you would like to view the salary on a monthly basis, click on the "Period" field once. The Period changes from "Year" to "Month," and the value in the "Salary" field is adjusted automatically to the monthly salary. If you de-select the "Exempt" button after it has been set, the heading for the "Salary" field changes from "Salary" to "Wage," this "Wage" field is cleared, and the "Period" field is reset to display "Hour."

You can save the current information into the fields at the bottom of the screen to keep track of past history. To do this click on the "Save Current Wage Data" button near the center of the screen.

The "To" and "Reason For Change" columns are left for you to fill out. In the *HyperCard* and *SuperCard* versions, you must click on the padlock in the upper-right corner to unlock the fields to make entry changes.

Benefits

There are two main sections to the Benefits screen (see Figures 10, 11, and 12). The upper section is for entering the types of benefits the employee has, the company supplying the benefit, the account number, and the date the employee started receiving the benefit. Benefits could include health insurance, life insurance, workers compensation, profit sharing or pension plan, and any other benefit your company may offer.

The lower section of this screen allows the entry of benefit transactions for a more detailed history. To enter a transaction, click on either the "Date" or "Code" fields. In the *HyperCard* and *SuperCard* versions we have implemented a special pop-up field of transaction code options from which the user can select (see Figures 13). In *LinkWay,* you simply enter these as text in the appropriate field as shown in Figure 12.

When you first use this section in *HyperCard* or *SuperCard* the "Code Options" field is blank. To enter new codes into the field unlock it by clicking on the padlock near the bottom of this field. Once you enter your codes, click on the padlock again to lock the "Code Options" field.

Now when you click on the options field, you select the option on which you click. To the left of the "Code Options" field is a small field for entering the date of the transaction. The current date is displayed by default. You may change this date by editing the field. Always use the HyperTalk and SuperTalk `short date` format (mm/dd/yy) for entering dates.

Once you have selected the option and date, you may click on the "Enter In Log" button. This places the date and selected option on the next available line in the log. You can then enter a detailed description for the transaction in the field next to the date and code entry. Keep all descriptions to a single line so they do not overlap into the next log entry.

In the *HyperCard* and *SuperCard* versions, if you make a mistake in your selection, you may edit the "Date" and "Code" fields by clicking on the padlock in

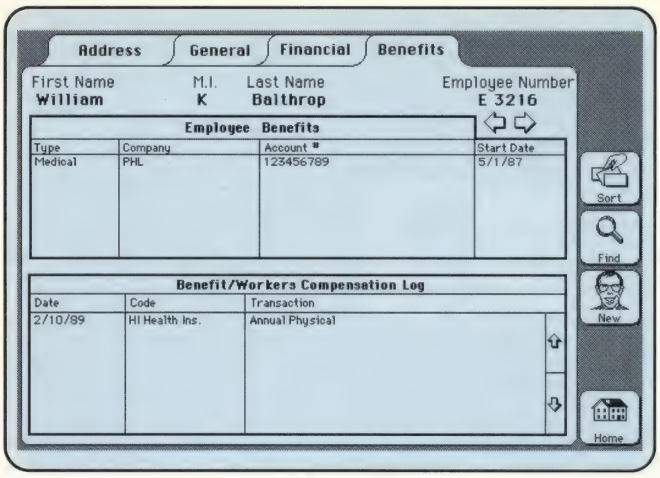


Figure 10

This is the HyperCard *Benefits screen.*

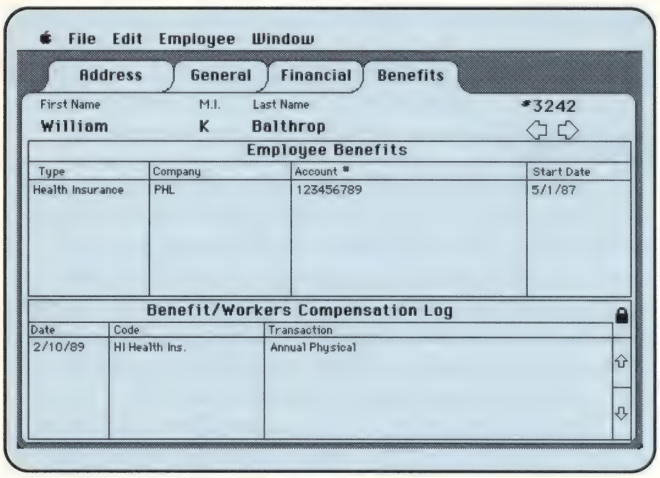


Figure 11

The SuperCard *version of the Benefits screen is, again, very much like the* HyperCard *version.*

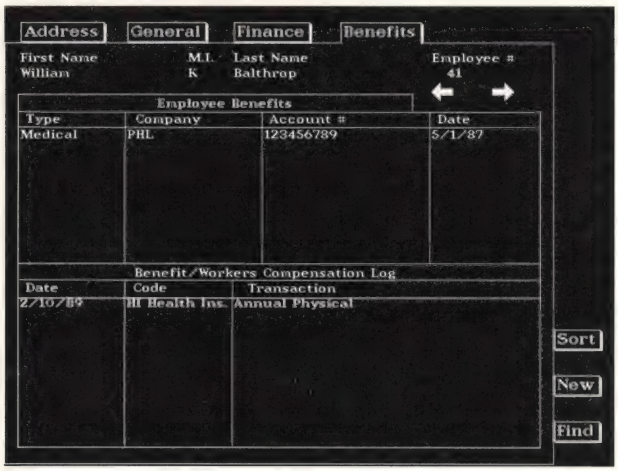


Figure 12

Benefits data is entered directly into the LinkWay *version.*

the upper-right corner of this section. To re-lock the fields, click on the padlock again. These fields should be kept locked to preserve historical information.

Special Features

There are several features implemented in each version that allow you to navigate or manipulate the

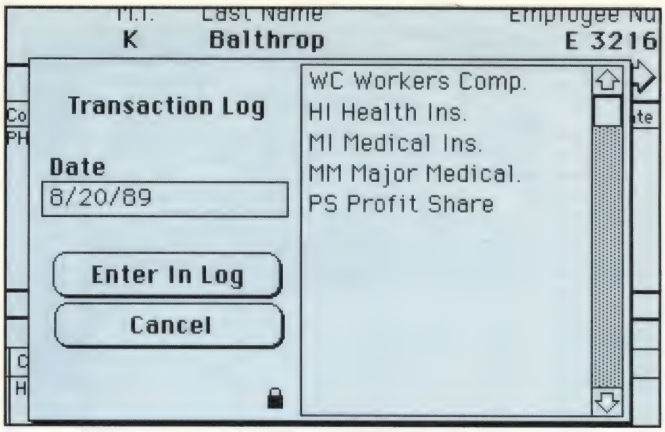


Figure 13

The HyperCard *and* SuperCard *versions use a specially coded transaction log which must be unlocked to be changed.*

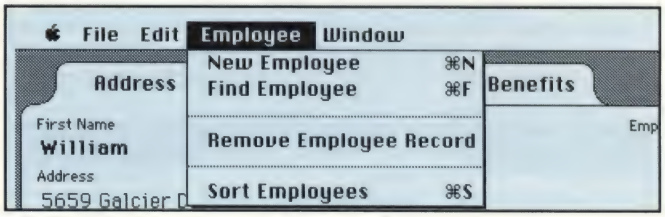


Figure 14

This is SuperCard's *Employee menu.*

employee records. The new record, find, remove, and sort features are implemented by buttons placed on the right side of the screen for the *HyperCard* and *IBM LinkWay* versions, and custom menus in the *SuperCard* version (see Figure 14).

New Employee Record

Selecting this option creates a new record . You are first prompted to enter the employee's first name, middle initial, and last name. Each of these should be entered separately as they are asked for. Once the name is entered new employee records are built for the four categories of information.

Remove Employee Record

This option erases an employee's records from the computer. Good record keeping practice should involve keeping old employee records around for a while. It is advisable to make a printout of any employee records before deleting them, you never know when you may need that information in the future.

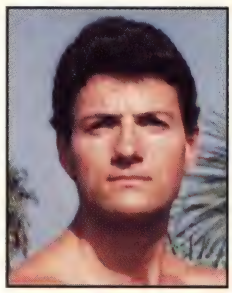
To remove an employee's records you must begin on that employee's Address screen. Select "Remove Employee Record" from the Employee menu in the *SuperCard* version, or click on the "Remove" button in the *HyperCard* and *LinkWay* versions.

Find Employee Record

After selecting "Find Employee" from the Employee menu in *SuperCard*, or clicking on the "Find" button in the *HyperCard* and *LinkWay* versions, you are prompted to enter the search string. This could be the

—continued on page 18

A Beginner's Tutorial on Creating...



Custom Menus in SuperCard

by Maurice Rizzuto

A great feature of *SuperCard* is that it enables you to easily create standard Macintosh menus, including the Apple menu, for your projects and standalone applications. This short tutorial will take you through the steps in creating your own custom menus.

The menus in Macintosh applications consist of a menu title to which the individual menu items are attached. In creating our menus, we'll first create the menu's title and give it a name, then create the separate menu items for that menu and assign a name and script to them. We'll be creating three menus—Apple (), File, and Goodies.

Begin by double-clicking *SuperEdit* from the Finder. At the "Open File" dialog box, click "New" to create a new project. Activate the project overview window by choosing Project "Untitled-1" from the Windows menu. Click on the menu icon in the project overview window (or choose Menus from the View menu). Note that our menus listing is empty (See Figure 1).

Now we are ready to create our first menu, which will be the Apple () menu. Choose "New Menu" (⌘-N) from the Edit menu. While the menu is selected, choose Menu Info (⌘-I) from the Edit menu. Type in "Apple" for the name in the Menu Info dialog and then click the "OK" button. With your Apple menu selected in the project overview window, choose "Open Menu" from the Edit menu (or press ⌘-O or double-click on the menu). Note that our Apple menu contains no menu items.

The Apple menu in Macintosh applications customarily contains an about-the-program command and a dotted line, below which the installed desk accessories are attached (look at *SuperEdit's* Apple menu for example). *SuperCard* makes it easy for you to create the standard Apple () Menu in your projects with all the desk accessories right where they belong.

Choose New Item (⌘-N) from the Edit menu to create a menu item. While the item is selected, choose "Item Info" (⌘-I) from the Edit menu. Type in "About my program..." for its name. Be sure to include the ellipses (...) in the name (created by pressing Option-;). The ellipses indicates to the user that a dialog box will be displayed when the menu item is selected.

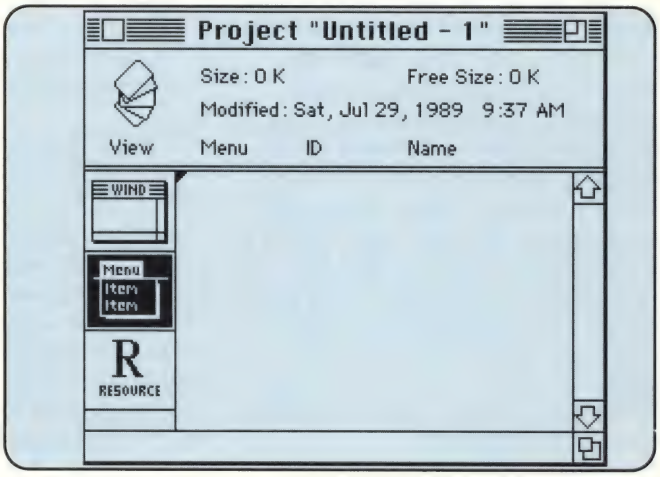


Figure 1
This is the "Project Overview" window with the menu icon selected before any menus have been created in a project.

Click the Script button to access the script window. Note that *SuperEdit* has automatically placed the appropriate handler, `on itemSelect...end itemSelect`, in the script window of the menu item. This handler is the one you will always use with menu items and is analogous to the `on mouseUp...end mouseUp` handler that is automatically placed in the script window of every new button. Let's enter a script that displays a small "About..." dialog when this command is selected while running the project. Go to the "Commands" pop-up menu within the script window and choose `answer`, then type "By Maurice" or whatever you want (enclosed within quotation marks). The script should read:

SuperCard *makes it easy for you to create the standard Apple () Menu in your projects with all the desk accessories right where they belong.*

Maurice Rizzuto, a senior member of the technical support staff of Silicon Beach Software, was the Beta Test Liaison for SuperCard, *and was heavily involved in its testing.*

```
on itemSelect
  answer "By Maurice"
end itemSelect
```

The line `answer "by Maurice"` is executed when the menu item is selected, hence the message handler's name, `itemSelect`. Close the script window by choosing "Close" from the File menu (or pressing ⌘-W or Enter, or clicking in its close box).

While in the item list window, choose "New Item" again to create another menu item. While it's selected, choose "Item Info" from the Edit menu. In the "Item Info" dialog, click in the "Simple Dividing Line" check box. This makes the name of the menu item a dotted gray line and also disables it. This menu item can't be selected but is merely a dotted gray line used to visually separate logical groups of menu items. We don't need to attach a script to this item, so click the "OK" button to close the "Item Info" dialog. Our Apple menu is now complete. You can test out this menu by playing with the mockup to the right of the regular *SuperEdit* menus (See Figure 2).

When your project is run, *SuperCard* will replace the word "Apple" with the Apple icon (🍎) and automatically attach your installed desk accessories below the simple dividing line.

Let's move on to creating our File menu. In Macintosh applications, the File menu contains commands for dealing with files (creating, opening, closing, saving, and printing them) as well as the "Quit" command. For our purpose, we'll include only two commands in our File menu: "Print" and "Quit." To create it we'll use the same basic technique that we used in creating the Apple menu.

While in the project overview window, with the menu icon selected, choose "New Menu" from the Edit menu. A new menu is created at the location of the insertion pointer. While the menu is selected, choose "Menu Info" from the Edit menu. Type in "File" for the name in the "Menu Info" dialog and then click the "OK" button. With the File menu selected in the project overview window, choose "Open Menu" from the Edit menu. Choose

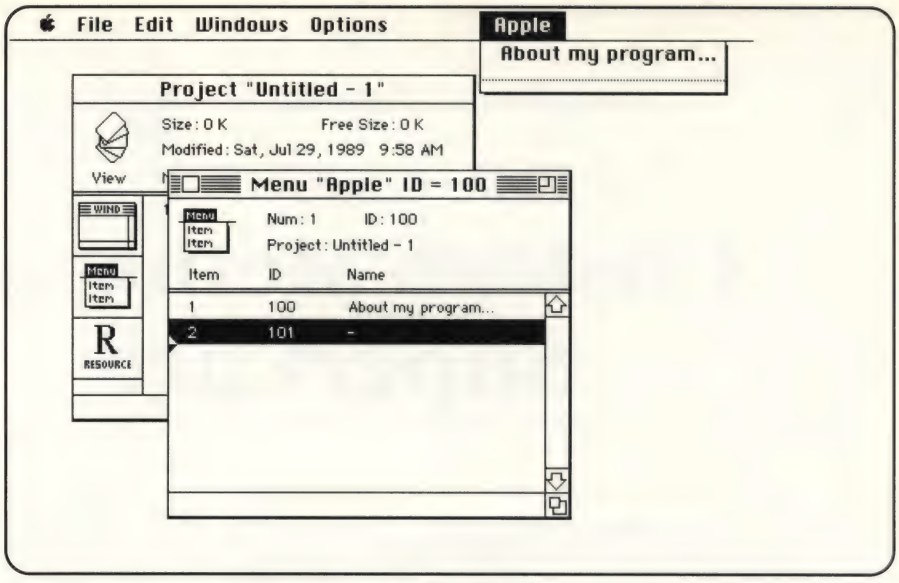


Figure 2

This is a SuperEdit *mockup of the Apple menu created by following the steps outlined here. When this project is run in* SuperCard *all desk accessories will automatically be added below the dashed line.*

"New Item" from the Edit menu to create a new item. While the item is selected, choose "Item Info" from the Edit menu. Type in "Print..." for its name. Let's assign a command key to this menu item: type "P" in the "Command Key" text field. Click the "Script" button. Enter this script:

```
on itemSelect
  print card
end itemSelect
```

When our project is run, choosing this menu item either via the mouse or by pressing ⌘-P will print the card in the top window. Close the script window.

Choose "New Item" from the Edit menu to create another menu item. While it's selected, choose "Item Info" from the Edit menu. Type in "Quit" for its name. Enter "Q" into the "Command Key" text field; this assigns the Command key shortcut. Click the "Script" button. Enter this script:

```
on itemSelect
  close all windows
end itemSelect
```

When *SuperCard* executes a `close all windows` command on the only open project, this causes *SuperCard* to quit. Close the script window. Our File menu is complete; play with the prototype on the right portion of the menu bar.

Lastly we'll create a Goodies menu with two items. From the project overview window, and while the menu icon is selected, choose "New Menu" from the Edit menu. While the menu is selected, choose "Menu Info" from the Edit menu. Type in "Goodies" for the name in the Menu Info dialog and then click the "OK" button. While the Goodies menu is still selected, choose "Open Menu" from the Edit menu. We'll create two menu items: "Age" and "Date." Choose "New Item" to create a new item. While it's selected, choose "Item Info" from the Edit menu. Type in "Age..." for its name. Click the "Script" button, enter the following script:

```
on itemSelect
  ask "How old are you?"
end itemSelect
```

Choosing this menu item when the project is run displays a dialog box asking "How old are you?" Close the script window.

To create our second item, choose "New Item" from the Edit menu. While the new menu item is selected, choose Item Info from the Edit menu. Name this item "Date..." and enter "D" for its command key. Click the "Script" button to access its script window, and enter the following script:

```
on itemSelect
  answer "The date is"&&the ¬
  long date&"."
end itemSelect
```

The second line of this script

displays the current date when you choose the "Date" menu item. Now close the script.

Our menus are now complete, so we need to create a script that instructs our project to insert our three menus when it's opened. This script needs to be placed in the Project Script. Activate the project overview window by choosing Project "Untitled-1" from the Windows menu. Choose "Project Script" from the Edit menu. (If the "Project Script" window contains the "Runtime Editor" script, choose "Select All" from the Edit menu and then press the delete or backspace key to clear the contents.) Enter the following script:

```
on startUp
  set lockMenus to true
  insert menu "Apple"
  insert menu "File"
  insert menu "Goodies"
  set lockMenus to false
end startUp
```

The line `set lockMenus to true` prevents the menu bar from updating. The next three lines insert the respective menus. As a menu is inserted into the menu bar, it is positioned to the right of any existing menus, thus the order in which you insert your menus is important. The line `set lockMenus to false` in the above script updates the menu bar and displays our three menus.

An alternate script for installing all the menus in a project when the project is opened is this:

```
on startUp
  set lockMenus to true
  repeat with x = 1 to the ¬
  number of menus
     insert menu x
  end repeat
  set lockMenus to false
end startUp
```

This script uses a repeat loop to insert all the menus that the project contains.

We've completed what we set out to do. Choose "Save" from the File menu to save your project. Now choose "Run" from the File menu.

This causes *SuperEdit* to quit and *SuperCard* to automatically open and run your project. When your project is opened, the three menus you created will be displayed in the menu bar. Try out the menu items and see how they work.

Peeking at Menu Item Scripts

When you are running a project that contains custom menus, you can peek at, and edit the scripts of menu items by pressing the Shift key and choosing the menu item. This is useful for making adjustments to your menu item scripts while you are running your project.

You can prevent users of your project from peeking at menu item scripts, by including the following script in the project script:

```
on startUp
  set the editMenus to false
end startUp
```

The editMenus property returns a value of either true or false. When it is true, you can peek at, and change, these scripts, when it is false, you can't.

Use the techniques discussed here to create your own custom menus in your *SuperCard* projects and standalone applications.

Employee Records

—continued from page 14

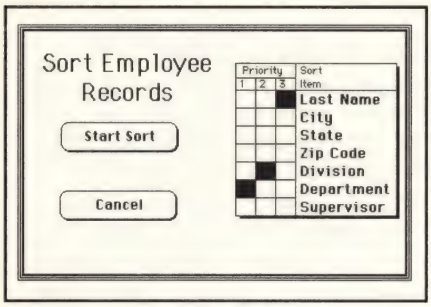


Figure 15

The Sort Employee Records screen allows sorts to be done in three priorities.

first or last name, or both. You could also search for last name and department. Simply separate the parameters with a space—e.g., if you search for "Smith Retail" while in the Address category, you will locate the first employee whose name is "Smith" and who has "Retail" entered into one of the other fields on the Address screen.

Find is restricted to the present category—i.e., if you are in the Financial category, the find can only be used on the information in that category. Note that each employee's name and number are in every category.

Programming "Employee Records"

The basic structure of all three versions is very similar. Each of the four category screens is kept in a separate *HyperCard* stack, *SuperCard* window, or *LinkWay* folder. Not only did this structure make construction of the applications easier, it also allows the user to browse around within a category without having to return back to the Address category every time a different employee is selected.

Sort Record

"Employee Records" provides you with a powerful sorting routine. By selecting your priorities, you can create nested sorts based upon several sort criteria. For example, you may sort the employee records first by last name, then by division within the departments, and finally by department.

The sort set up screen (Figure 15 shows the *HyperCard* version) allows you to select three sort parameters from any of seven data fields: Last Name, City, State, Zip Code, Division, Department, and Supervisor. Select the parameters desired by clicking on the box to the left of the field name. There can only be one sort parameter for each level of the sort. A parameter may only be selected once in the sort priority. You may sort on one, two, or all three parameters. Sort priority one must be selected, but priorities two and three are optional.

Listing 1 - Make Selection Script for HyperCard and SuperCard

```
on MakeSelection
  global Sort1,Sort2,Sort3
  get the target
  put the short name of it into Tbtn
  put char 2 of word 1 of Tbtn into Row
  put char 2 of word 2 of Tbtn into Column
  if hilite of btn Tbtn is true then
    set hilite of btn Tbtn to false
    if Column is 1 then put empty into Sort1
    if Column is 2 then put empty into Sort2
    if Column is 3 then put empty into Sort3
    exit MakeSelection
  end if
  if (char 1 of Tbtn is "R") ¬
  and (Column is in "1234567") then
    if Column is 1 then
      if Sort1 is not empty¬
      then set hilite of btn ("R"&Sort1&&"C1") to false
      if Sort2 is Row then set hilite of btn ¬
      ("R"&Row&&"C2") to false
      if Sort3 is Row then set hilite of btn ¬
      ("R"&Row&&"C3") to false
      set hilite of btn Tbtn to true
      put Row into Sort1
    else
      if Column is 2 then
        if Sort2 is not empty¬
        then set hilite of btn ("R"&Sort2&&"C2") to false
        if Sort1 is Row then set hilite of btn ¬
        ("R"&Row&&"C1") to false
        if Sort3 is Row then set hilite of btn ¬
        ("R"&Row&&"C3") to false
        set hilite of btn Tbtn to true
        put Row into Sort2
      else
        if Column is 3 then
          if Sort3 is not empty¬
          then set hilite of btn ("R"&Sort3&&"C3") to ¬
          false
          if Sort1 is Row then set hilite of btn ¬
          ("R"&Row&&"C1") to false
          if Sort2 is Row then set hilite of btn ¬
          ("R"&Row&&"C2") to false
          set hilite of btn Tbtn to true
          put Row into Sort3
        end if
      end if
    end if
  end if
end MakeSelection
```

HyperCard and *SuperCard* process the sort in the same manner, however, because *IBM LinkWay* does not have a built in sort command it is handled with the DOS "Sort" routine.

HyperCard/SuperCard

The *HyperCard* and *SuperCard* versions use the same script to select the sort parameters. Listing 1 contains this script.

The `openCard` routine clears the sort priorities entries.The `MakeSelection` handler is called

Listing 2 -The HyperCard "Sort" button script

```
on mouseUp
  global Sort1,Sort2,Sort3
  if Sort1 is empty then
    answer "You havn't selected a primary sort item!"
    exit mouseUp
  else put line Sort1 of card field "Sort List" into ¬
  SortField1
  if Sort2 is not empty then put line Sort2 ¬
  of card field "Sort List" into SortField2
  show Msg
  set lockscreen to true
  set the lockmessages to true
  -- Sort Third Field
  if Sort3 is not empty then
    put "Sorting Address 3!" into Msg
    put line Sort3 of card field "Sort List" into ¬
    SortField3
    go "Employees"
    sort by field SortField3
  else go "Employees"
  -- Sort Second Field
  if Sort2 is not empty then
    put "Sorting Address 2!" into Msg
    sort by field SortField2
  end if
  -- Sort Primary Field
  put "Sorting Address 1!" into Msg
  sort by field SortField1
  -- Set up key sort fields in other windows
  put "Setting Up Data Windows!" into Msg
  repeat with x=2 to the number of cards
    go cd x of "Employees"
    put field "Employee Number" of card x into ENumber
    go cd Enumber of "Employee General"
    put x into field "Sort Key"
    go cd Enumber of "Employee Financial"
    put x into field "Sort Key"
    go cd Enumber of "Employee Benefits"
    put x into field "Sort Key"
  end repeat
  -- Sort General, Financial and Benefits windows
  go "Employee General"
  put "Sorting General!" into Msg
  sort by field "Sort Key"
  go "Employee Financial"
  put "Sorting Financial!" into Msg
  sort by field "Sort Key"
  go "Employee Benefits"
  put "Sorting Benefits!" into Msg
  sort by field "Sort Key"
  put "Sorting Complete!" into Msg
  pop card
end mouseUp
```

from each of the 21 buttons which make up the priority selection area. The only script in these buttons is:

```
on mouseUp
  MakeSelection
end mouseUp
```

This handler ensures that only one parameter is selected for each priority and that the `Sort1`, `Sort2`, and `Sort3` variables are set to the currently selected row. These global variables are used in the sort routine.

The *HyperCard* version of the sort routine is in Listing 2. The *SuperCard* routine differs only because the *HyperCard* version uses multiple stacks, and the *SuperCard* version uses a single project with multiple windows. Just keep in mind when working in *SuperCard* that the `go` command in the Listing should refer to a `window` instead of a stack.

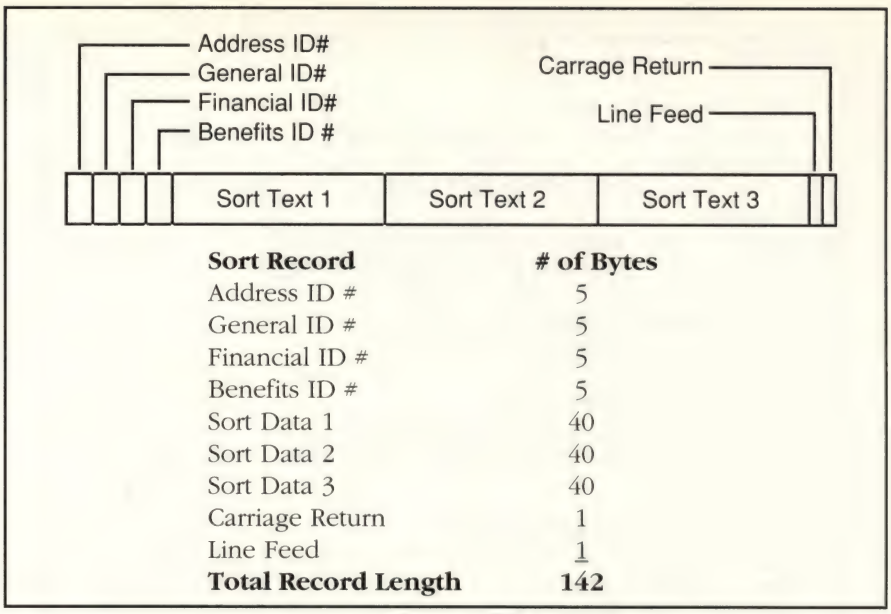
After the sort is complete the `pop card` command returns the user to the card where the sort was first requested.

The *IBM LinkWay* Sort

The *IBM LinkWay* version has one major difference from the other two versions because *LinkWay* does not currently support the sorting of pages. We felt it was important to provide the sort feature, even though this does make the *LinkWay* version more complex than the other versions.

The *LinkWay* sort setup is handled in a very similar fashion to the *HyperCard* and *SuperCard* versions. The screen looks quite similar to Figure 15, except that instead of highlighted buttons, the *LinkWay* version uses asterisks to mark each option. Here is the script of the button that does the priority selecting:

```
Var Resp(2);
Var Row(1),Column(1),Old
(2),New(2),Temp(42);

if sort1<1 set sort1=0;
if sort2<1 set sort2=0;
if sort3<1 set sort3=0;

Menu Resp,15,
  "Select An Item:":
  "  Last Name    ":
  "  City         ":
  "  State        ":
  "  Zip Code     ":
  "  Division     ":
  "  Department   ":
  "  Supervisor   ";

if Resp<2 jump End;
set Row=Resp-1;

Menu Resp,18,
  "Select Sort Order:":
  "        1         ":
  "        2         ":
  "        3         ";

if Resp<2 jump End;
set Column=Resp-1;

set Temp=Options;
if Column=1
  { if Sort1>0 set Temp(
(Sort1-1)*6+1)="-";
    set Temp((Row-1)*6+1)="*";
    if Sort2=Row set Temp(
(Sort2-1)*6+3)="-";
    if Sort3=Row set Temp(
(Sort3-1)*6+5)="-";
    set Sort1=Row; }

if Column=2
 { if Sort2>0 set Temp((Sort2-
1)*6+3)="-";
    set Temp((Row-1)*6+3)="*";
    if Sort1=Row set Temp(
(Sort1-1)*6+1)="-";
    if Sort3=Row set Temp(
(Sort3-1)*6+5)="-";
    set Sort2=Row; }

if Column=3
  { if Sort3>0 set Temp(
(Sort3-1)*6+5)="-";
    set Temp((Row-1)*6+5)="*";
    if Sort1=Row set Temp(
(Sort1-1)*6+1)="-";
    if Sort2=Row set Temp(
(Sort2-1)*6+3)="-";
    set Sort3=Row; }

set Options=Temp;

@End;
```



Sort Record	# of Bytes
Address ID #	5
General ID #	5
Financial ID #	5
Benefits ID #	5
Sort Data 1	40
Sort Data 2	40
Sort Data 3	40
Carriage Return	1
Line Feed	1
Total Record Length	**142**

Figure 16

This is the structure of the LinkWay *sort record for the routine used in "Employee Records."*

In order to facilitate moving from record to record in a sorted fashion, we built an index list of employees. This list contains all four page ID numbers for an employee's four categories of information, as well as up to three sort criteria. The structure of a sort record is shown in Figure 16.

After the sort criteria are selected a text file is created to be used as an index file. The index is sorted using the DOS "SORT.EXE" routine which is provided on the PC-DOS system disk. The sort criteria allows us to begin at any character in the index text file. By starting at the 21st character we ignore the page ID numbers in our sort, but maintain them with each record. After the sort is complete, the employee records are updated. Each page in a category contains page ID pointers to the previous and next employee in sorted order. The ID numbers for that employee's records in the other three categories are also stored (see Figure 17).

Two major shortcomings in *IBM LinkWay* are its limited script editor with few editing features, and the fact that scripts cannot exceed 3K in size. You can overcome both problems by writing your scripts in *LWEdit*. Although it is a simple text editor, it has many features not found in *LinkWay's* built-in script editor. To use *LWEdit* you must create a "Document" button and link it to a text file. Next, to execute the text file as a script, put two lines of code in the "Script" button used to call this routine. For example, to call a script "SORTP2.SCR" the code in the script button is:

```
Load Sp2,"Sortp2.scr";
script Sp2;
```

This `Load` command places the entire contents of the file SORTP2.SCR into the variable `Sp2`. The `script` command then causes the contents of the variable `Sp2` to execute as script.

The second problem we mentioned is the fact that *LinkWay's* interpreter can only work with 3K

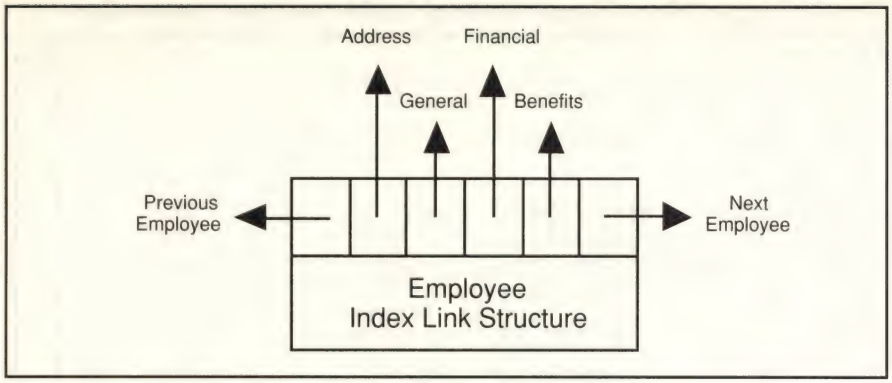


Figure 17

chunks of script. This limit is easily reached when you write even slightly complex scripts. The solution is to break your scripts up into 3K or less chunks and use *LWEdit*. You can then link them together by placing two lines of code at the end of the script chunk, which will Load and execute the next chunk with the `script` command. The two lines of code listed just above do that for the sort routine. By placing these two lines at the bottom of the *LWEdit* file "SORT.SCR" the two scripts become linked together.

There are several things to keep in mind if you use this technique. Once you execute a script with a script command you are on a one-way street. There is no way to return to the calling routine. When breaking up your script you should be careful to keep jumps and the labels they jump to all in the same chunk of script.

The scripts below are responsible for sorting the "Employee" folder, and reorganizing the links to the "General," "Financial," and "Benefits" folders. Notice how the first script calls the second using the `script` command.

LinkWay SORT.SCR:

```
if Sort1<1 jump End;
if Sort1>7 jump End;
set Rec=" ";
noshow;
DOS "IF EXIST EMPSORT.SRT DEL
EMPSORT.SRT";
set Rec(141)=chr 13;
set Rec(142)=chr 10;

set Emp=1;
link Return,"Employee";
if Count<3
  { msg "You don't have
enough","records to sort!";
    link 0,"SortScrn";
    jump End; }

@Loop;
go Emp+1;
set Num=EmpNum;
set Links=LinkList;
set Rec(1,20)=Links(6,20);

if Sort1=1 set Rec
(21,40)=Last;
if Sort1=2 set Rec
(21,40)=City;
if Sort1=3 set Rec
(21,40)=State;
if Sort1=4 set Rec(21,40)=Zip;
if Sort1=5 set Rec
(21,40)=Division;
if Sort1=6 set Rec
(21,40)=Depart;
if Sort1=7 set Rec
(21,40)=Supervis;

if Sort2=1 set Rec
(61,40)=Last;
if Sort2=2 set Rec
(61,40)=City;
if Sort2=3 set Rec
(61,40)=State;
if Sort2=4 set Rec(61,40)=Zip;
if Sort2=5 set Rec
(61,40)=Division;
if Sort2=6 set Rec
(61,40)=Depart;
if Sort2=7 set Rec
(61,40)=Supervis;

if Sort3=1 set Rec
(101,40)=Last;
if Sort3=2 set Rec
(101,40)=City;
if Sort3=3 set Rec
(101,40)=State;
if Sort3=4 set Rec
(101,40)=Zip;
if Sort3=5 set Rec
(101,40)=Division;
if Sort3=6 set Rec
(101,40)=Depart;
if Sort3=7 set Rec
(101,40)=Supervis;

set Pos=(Emp-1)*142;
write "Emp-
Sort.SRT",Pos,142,Rec;
set Emp=Emp+1;
if Emp<Count jump Loop;

DOS "Echo Off";
DOS "SORT <EmpSort.SRT/+21
>SortTemp.SRT";
DOS "Copy SortTemp.SRT Emp-
Sort.SRT";
DOS "Echo On";

set Size=fsize "EmpSort.SRT";
if Size<142
  {Msg="Sort Can Not Be
Completed";
   jump End; }

set Emp=1;
read"EmpSort.SRT",0,20,Irec;
set Current=Irec;
if Size>142
  { read "Emp-
Sort.SRT",142,20,Irec;
    set Next=Irec; }
linkCurrent(1,5),"Employee";
set Links=LinkList;
set Links(1,5)="*";
set Links(26,5)=Next(1,5);
set LinkList=Links;

linkCurrent(6,5),"EmpGenrl";
set Links=LinkList;
set Links(1,5)="*";
set Links(26,5)=Next(6,5);
set LinkList=Links;

linkCurrent(11,5),"EmpFin";
set Links=LinkList;
set Links(1,5)="*";
set Links(26,5)=Next(11,5);
set LinkList=Links;

linkCurrent(16,5),"EmpBen";
set Links=LinkList;
set Links(1,5)="*";
set Links(26,5)=Next(16,5);
set LinkList=Links;
set Prev=Current;
set Current=Next;
if size<285 jump Last;

@Replace;
read "EmpSort.SRT",
(Emp+1)*142,20,Next;
linkCurrent(1,5),"Employee";
set Links=LinkList;
set Links(1,5)=Prev(1,5);
set Links(26,5)=Next(1,5);
set LinkList=Links;

linkCurrent(6,5),"EmpGenrl";
set Links=LinkList;
set Links(1,5)=Prev(6,5);
set Links(26,5)=Next(6,5);
set LinkList=Links;

linkCurrent(11,5),"EmpFin";
set Links=LinkList;
set Links(1,5)=Prev(11,5);
set Links(26,5)=Next(11,5);
set LinkList=Links;
```

```
linkCurrent(16,5),"EmpBen";
set Links=LinkList;
set Links(1,5)=Prev(16,5);
set Links(26,5)=Next(16,5);
set LinkList=Links;

set Prev=Current;
set Current=Next;
set Emp=Emp+1;
if Emp<Count-2 jump Replace;

@Last;
load Sp2,"Sortp2.scr";
script Sp2;
@End;
```

LinkWay SORTP2.SCR:

```
linkCurrent(1,5),"Employee";
set Links=LinkList;
set Links(1,5)=Prev(1,5);
set Links(26,5)="*";
set LinkList=Links;

linkCurrent(6,5),"EmpGenrl";
set Links=LinkList;
set Links(1,5)=Prev(6,5);
set Links(26,5)="*";
set LinkList=Links;

linkCurrent(11,5),"EmpFin";
set Links=LinkList;
set Links(1,5)=Prev(11,5);
set Links(26,5)="*";
set LinkList=Links;

linkCurrent(16,5),"EmpBen";
set Links=LinkList;
set Links(1,5)=Prev(16,5);
set Links(26,5)="*";
set LinkList=Links;

read"EmpSort.SRT",0,20,Irec;
link 0,"Employee";
show;
linkIrec(1,5),"Employee";
```

Scanning Pictures

The Address screen for each employee has space reserved for a scanned graphic of that employee. The methods available for getting a scanned graphic of your employee into one of the Employee Records versions varies from one to the next. We describe several options below.

HyperCard— Hand-held or Live

One alternative for *HyperCard* is the ProVis video scanner. This method allows you to directly scan your live employee into your stack using a video camera. ProVis has licensed the "HyperScan" XCMD from Apple and has produced its own "HyperVision" stack to make scanning into *HyperCard* easy. By making changes to the ProVis "HyperVision" stack we can automatically paste the scanned image into our "Employee" stack.

Listing 3 - "Scan" Button in HyperCard "Employee Records"

```
on mouseUp
 global Where,xStart,yStart,xSize,ySize
  get the clickloc
  set locktext of me to false
  click at it
  click at it
  put the selectedtext into Scanner
  set locktext of me to true
  get the rect of bg btn "Scan Image"
  put item 1 of it into xStart
  put item 2 of it into yStart
  put item 3 of it into xEnd
  put item 4 of it into yEnd
  put xEnd-xStart into xSize
  put yEnd-yStart into ySize
  hide btn "Cancel"
  hide card fld "Scanner List"
  hide card fld "Cover"
  choose pencil tool
  drag from xStart,yStart to xStart+2,yStart+2
  choose select tool
  drag from xStart,yStart to xEnd,yEnd
  doMenu "Cut Picture"
  choose browse tool
  put "Employee" into where
  push card
  if Scanner is "ProVis" then
    go cd "copy Stand" of stack "HyperVision.Mod"
  else if Scanner is "ScanMan" then
    show menubar
    doMenu "ScanMan"
  else if Scanner is "LightningScan" then
    show menubar
    doMenu "LightningScan"
  end if
end mouseUp
```

You can also use a hand-held scanner to get images of employees into your stack. Two hand-held scanners are currently available for the Macintosh: Logitech's ScanMan and ThunderWare's LightningScan. These scanners are most easily employed from within *HyperCard* when installed as desk accessories and accessed through the Apple (🍎) menu on the left side of the menu bar. If your menu bar is not showing, display it by pressing ⌘-space.

We found ThunderWare's LightningScan the easiest to use from *HyperCard*. Start the LightningScan desk accessory and a new menu is added to the menu bar called "Scanner." Set your scanner to 100 dots per inch, and the dot size to the smallest dot setting. Select "Scan" from the Scanner menu, then scan the image you want. Once the image is scanned, click on the "Done Scan" button. Now use the selection tool to copy a section of the scanned image. Close LightningScan and you are back at your *HyperCard* card. Paste the graphic onto the card layer of your card, then use the selection tool to move the graphic into place.

Logitech's ScanMan works very well on the IBM PC, however ,the software on the Macintosh is not suited to doing the 72 dots per inch (dpi) graphics supported by *HyperCard*. After scanning an image at 100 dpi, ScanMan's software does a compression of the image which causes major distortions. The most efficient way we found to get a clean image into *HyperCard* from the ScanMan hand-held scanner is to select an option called "Full Size" from the ScanMan menu. This displays the image without the compression. The only way to get this

Listing 4 - Modified "Scan Area" button script for " HyperVision.mod"

```
on mouseDown
  global where, xSize, ySize
  hide me
  if where is not empty then
    hide bg btn "Save Area"
    get the rect of me
    put item 3 of it - item 1 of it into oldX
    put item 4 of it - item 2 of it into oldY
   put "32,27,"&32+xSize&","&27+ySize into pinRect
   get HyperScan("DragRect",rect of me,pinRect)
    set the rect of me to it
    put item 1 of it into item 1 of newRect
    put item 2 of it into item 2 of newRect
    put xSize * (item 3 of it - item 1 of it) div 512 ¬
    into newX
    put newX + item 1 of newRect into item 3 of newRect
    put ySize * (item 4 of it - item 2 of it) div 342 ¬
    into newy
    put newY + item 2 of newRect into item 4 of newRect
    set rect of bkgnd Button "Save Area" to newRect
    show bg btn "Save Area"
  else
    put "32,27,201,161" into pinRect
    get HyperScan("DragRect",rect of me,pinRect)
    set the rect of me to it
  end if
  show me
  scalegauge
end mouseDown
```

image into *HyperCard* is to save the entire screen by pressing ⌘-Shift-3. This creates a *MacPaint* style bitmap graphic of the current screen called "Screen 0." Return to *HyperCard* by closing ScanMan. Then import the "Screen 0" file into *HyperCard* with the "Import Paint" menu option. This works fine when you are on a small screen Macintosh like a Mac Plus or an SE. If you are on a large screen on a Mac II, however, the picture is not saved in the correct format for *HyperCard*—instead it is rotated ninety degrees to the left. It is necessary to import the "Screen 0" picture into a paint program like *SuperPaint*, or *Studio/1* and rotate it 90 degrees to the right, then import it into *HyperCard*.

Other scanners may be used. Even if your scanner does not have a *HyperCard* interface, you can still copy and paste the image from your scanner software, or a paint program, into the *HyperCard* stack. Simply copy the image you want onto the clipboard, find the employee record you want to paste the image into, then paste the image from the clipboard and position it in the space provided, and trim it to fit using *HyperCard's* Paint tools.

Using ProVis's "HyperVision"

"HyperVision" is a product produced by Pixelogic to be used with their ProVis video scanner. We have made a few modifications to "HyperVision" which will allow another stack to enter it, scan a picture, then return to the calling stack and automatically paste the scanned picture. Since we could not distribute this modified stack, we present you with all of the modifications necessary, and instruction on how to make these modifications yourself. You should be relatively knowledgeable about editing HyperTalk before attempting these modifications. [Changes similar to these were originally presented in *HyperLink Magazine* Vol. 2, #3.—Ed.]

First, take a look at Listing 3, the script in the background field "Scanner List" in the "Employee" stack which calls the scanners.

This script is placed in a locked field that contains a list of available scanners. The list currently includes the ProVis video scanner, ScanMan hand-held scanner, and ThunderWare's LightningScan hand-held scanner.

If the ProVis scanner is selected, the process of pasting the scanned graphic into the "Employee" stack is automated. The first step in automating this process is setting up four global variables.

`XStart` X coordinate for upper-left corner of graphic area
`YStart` Y coordinate for upper -eft corner of graphic area
`XSize` Width of graphic area in pixels
`YSize` Height of graphic area in pixels.

Once these variables are set the scripts take you to a modified "HyperVision" stack. When you are finished scanning, you are returned to the "Employee" stack, and the scanned graphic is automatically pasted into place.

"HyperVision" Modifications

Make a backup of your "HyperVision" stack by opening "HyperVision," and selecting "Save a Copy" from the File menu. Enter "HyperVision.Mod" as the name of the new stack. Next open it by choosing "Open" from the File menu so all modifications will be made in the "HyperVision.Mod" stack.

Go to the "Copy Stand" card. Select the button tool from the Tools palette. Select the "Scan Area" button in the upper left corner of the screen. Copy this button by selecting "Copy Button" from the Edit menu. Paste a new copy of this button by selecting "Paste Button" from the Edit menu. Double-click on this new button to bring up the button dialog box. Change the name of the button to "Save Area." Replace the button script with the follow script:

```
on mouseDown
  send mouseDown to bg btn ¬
  "Scan Area"
end mouseDown
```

Go to the Video Scan card and repeat this procedure for the "Scan Area" button on that card. Next change the script in the background button "Scan Area" on this card and the "Copy Stand" card to that shown in Listing 4.

For the final modification to this stack go to the "Halftone" card. Change the script of the "Save" button to read as follows:

```
on mouseDown
end mouseDown

on mouseUp
  global where, xStart
  global yStart, xSize, ySize
  if where is not empty then
    put empty into where
    domenu compact stack
    choose select tool
    drag from 0,0 to ¬
    xSize,YSize with optionKey
    doMenu "Copy Picture"
    pop card
    choose select tool
    doMenu "Paste Picture"
    drag from 0,0 to ¬
    xStart,yStart
    choose browse tool
    hide menuBar
  else
    set hilite of me to true
    set cursor to busy
    saveCard
    set hilite of me to false
  end if
end mouseUp
```

When in use, the "Save Area" button is sized and positioned to capture the picture area desired.

SuperCard

As you read this, utilities are being developed to make scanning images directly into *SuperCard* easier. However, until those utilities are made available you need to rely on using the Copy and Paste functions to move graphics onto your cards.

To place a scanned image into your "Employee Records" project, you first need to create the employee record. Then exit *SuperCard* and enter the paint program or scan program. Copy the graphic you want to use in the Employee project. Re-enter the "Employee Records" project in *SuperCard* and find the new employee record. Paste the image onto the screen. Once the image is on screen and selected with the Selection tool, you can position it and trim it to size. We have included the standard *SuperCard* Paint palette in the "Employee Records" project to make this as easy as possible.

IBM LinkWay

There are a number of scanners available for the IBM line of computers. As of this writing there is no way to directly use these scanners from within *IBM LinkWay*. You can, however, use *LWCapture*, which is available as part of the *IBM LinkWay* package, to save screens in a format *LinkWay* will recognize. You must be in MCGA 2 color high resolution mode when you scan graphics for the "Employee" folder.

To use *LWCapture* it must first be installed. to install *LWCapture* from DOS ensure that the program file is available in the current DOS PATH , then type "LWCaptur" at the DOS prompt and press Enter. *LWCapture* is a "terminate and stay resident" (TSR) program and will be installed until you either turn off your computer or reboot by pressing Control-Alt-Delete.

We reviewed Logitech's ScanMan hand-held scanner and *PaintShow Plus* software. The only problem we encountered was when we tried to use *LinkWay* as the "shell" program and go to DOS via the "Dos Cmd" option of the Options menu. The present version of LinkWay keeps a large part of memory reserved, and the *PaintShow Plus* scanning area was limited to a very small size even at 100 dots per inch (dpi). For best results we found it best to completely exit *LinkWay*.

Here's the process that worked for us. From DOS, with *LWCapture* installed, run *PaintShow Plus* and scan your image. When the graphic is displayed on the screen, press Shift-PrtScn. A prompt at the top of the screen asks for a file name for this screen image. Enter a name for the file and press Enter. to save the image to disk.

LinkWay has a special object type called a "Picture." We placed one of these for an employees picture as shown in Figure 3. On a new employee screen, this picture object does not contain anything. You must tell it which picture to display.

To do this, first ensure that the Access Level in *LinkWay* is set to "Format." Click the mouse in the upper-left corner of the screen to access the The "Folder" menu. Then select "Access Level" from the menu. You may be asked for a password at this point. We always leave the password blank so anyone can change the Access Level. You may wish to enter a password to protect your folders, anyone who needs to change add a picture must know this password. A pop-up menu is displayed in the center of the screen. Click on the bottom option "Format," then click on the close dot in the upper-left corner of this menu. This places *LinkWay* into "Format" mode.

Click once in the area where the picture will be placed. The picture object becomes outlined. Now click on the Object menu option — the third from the left—from the main menu bar. This displays a list of options which can be performed on this object—click on the "Edit" option. A menu appears in the center of the screen with a list of graphic files compatible with the mode being used. If the file you created is not in this list, then it is either in a different directory, or its graphic mode is not compatible. The graphic you created must have a .PCH extension. If it has a different extension, then you were not in the proper mode when you captured the scanned graphic.

With the graphic file listed in the Picture File menu, select that file by clicking on it. Then close the menu by clicking on the close dot. The graphic then appears inside the picture object. A new menu then appears in the upper-left corner of the screen which allows you to adjust the alignment of the graphic within the picture window. Click on one of the options to move the graphic within the window. Click outside of the "Adjust" menu when you are done.

To use a different scanner with *LinkWay*, run the scanner software which accompanies your scanner and allows the scanned image to be displayed on the screen. Scan the image you want to use and display it on the screen. When doing this, make sure that the current screen mode you are using to display the scanned image is the same mode you wish to use in your *LinkWay* folder. The "Employee" folder is written in high resolution, 2 color MCGA mode—i.e., the DOS extension for the folder is .LWH, and .PCH is the extension of all the graphics files). Then follow the procedure outlined above to make the picture the proper format using *LWCapture*.

Linking Away

How to Create a General Purpose Data Management Folder in IBM LinkWay

by Larry Burtness

Folders for individual data management tasks are common applications for *LinkWay*, from personal mailing lists to collection inventories. These applications are often similar in function and form, with common techniques for creating different applications.

The folder we will develop in this the first "Linking Away" column is for keeping information about a compact disc collection. It includes fields for data, fields for labels, and buttons for carrying out specific actions such as adding or deleting records from the file, printing a page, and browsing from page-to-page.

These buttons will be common to this type of folder, and the scripts in the buttons can be modified to accommodate many different types of applications.

General Folder Organization

The "CDFILE" folder is organized so that the base page contains the field labels and all of the buttons, with data pages keeping information about compact discs. Each data page will have fields for a disc title, the artist, and ten songs, with their playing times.

When creating this kind of folder, it helps to start by listing on a piece of paper all of the data fields that will be used to store information and the corresponding fields which will be used to label the data fields. Also sketch the general screen appearance. Fields to be used as labels will be on the base page and will show through to all of the data pages and serve as labels for every page in the folder. The fields that contain data, however, must be placed only on the data pages.

These fields that are named on the data page. Field names were chosen which allow easy reference to the fields from the "ADD REC" button. The fields are named "TITLE," "ARTIST," "S1" through "S10," and "T1" through "T10." All of these fields are created on page 1, the first data page in the folder, and the LinkWay `clone` command is used to create a new page for each CD.

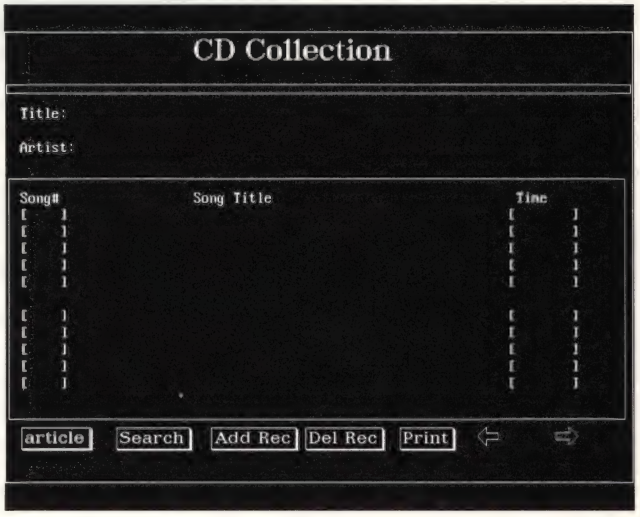


Figure 1
This screen from the MCGA mono mode version of our "CDFILE" LinkWay *folder shows the base page alone with no data.*

The buttons across the bottom of the screen, all placed on the base page, allow the user to conduct a keyword search, add a new page to the folder, delete a page, print a page, and browse forward and back through the folder.

Creating the Folder

Select "New" from the *LinkWay* Folder menu, then type in a name for the new folder—"CDFILE." A blank screen appears with the status line at the bottom indicating that you are on the base page of the folder.

The fields for the labels are placed on the base page first. Figure 1 shows all of the objects (fields and buttons) that are visible on the base page. Select "New" from the Object menu and create fields for labels in the positions shown on Figure 1. When you create a new field, immediately fill it with some text. This allows the otherwise invisible fields to be easily found on the screen. The field will not otherwise be visible and you may lose track of where you have placed fields on the page if they are all invisible.

In addition, if there are several fields that have the same characteristics such as size, font, and color, it is

Larry Burtness is the Director of the Educational Technology Center for Educational Service District #189 in Mount Vernon, Washington. He specializes in training teachers to apply computers in education. He has been teaching HyperCard *since its introduction, and using pre-release versions of* IBM LinkWay *for over a year.*

quickest to create one field, use the Object menu's "Cut" command to put it in a scrap object file, then paste it in place using the Object menu's "Paste" command rather than going through the complete field creation sequence. A field copy is pasted in the original's location and must be moved to its proper place on the page.

After all of the base page fields are created, you are ready to make a new page as the first data page. Again, the process of pasting a copy, then moving the copied field was done for the fields "S1" through "S10" and the fields "T1" through "T10" as shown in Figures 2 and 3. After moving them to the proper locations, each is edited to assign each field's unique name, and data should be entered so the field's location is easily identified.

Do all of the folder creation work entirely on either the base page or page 1. The "ADD REC" button will allow more pages to be added later as they are needed, but it uses the original page 1 as a template, thus this page should be completed before this button is activated.

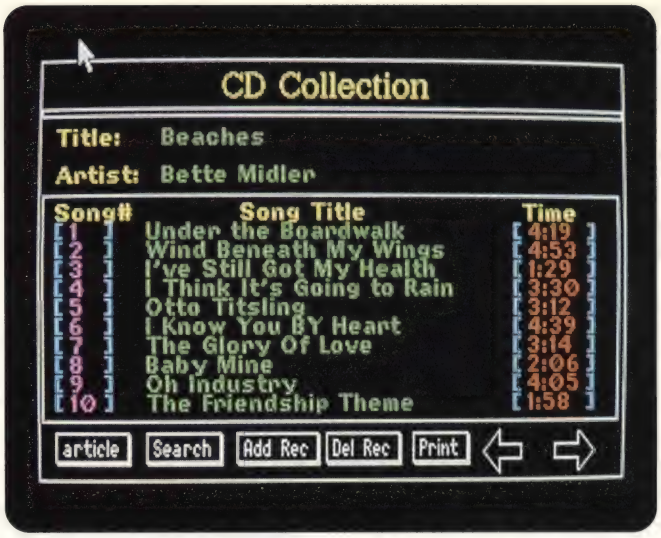


Figure 2

This screen shot shows an example of a completed data record for the "CDFILE" LinkWay *folder using MCGA color mode with 256 colors.*

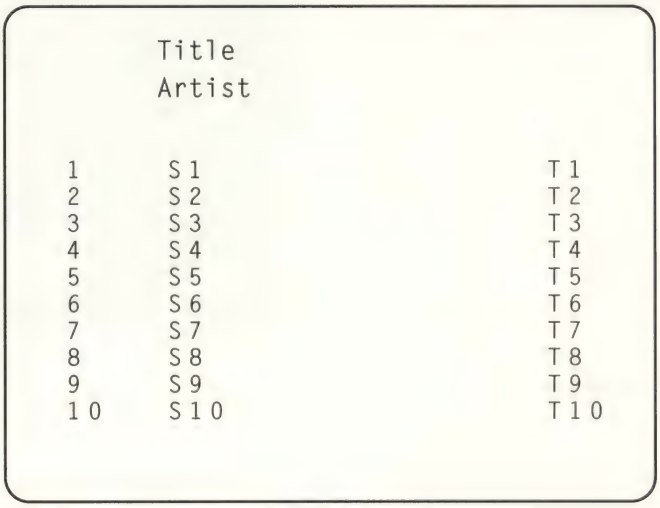


Figure 3

This shows the names for each of the fields on the data page. It is critical that the correct names are used for the scripts shown here to operate properly.

Base Page Buttons

Many operations, such as adding records, should be available at all times in the folder. Such global operations should be placed in buttons on the base page. Other operations specific to pages should be placed in buttons directly on the data page where they are used.

The arrow buttons for browsing forward and back are "Go" buttons placed using the appropriate icons. The left arrow indicates a go previous page and the right arrow a next. "Go" buttons automatically wrap from the last page in the folder to the first page when the end of the folder is reached. All of the other buttons used in this example are script buttons.

"Search" Button

The "Search" button is a simple example of a keyword search button. It asks the user for input of a keyword and then searches for the word on the pages in the folder. To create the button, select "New" from the Object menu, then choose "Button" in the pop-up menu. Next name the button "Search," select the shadowed box, and enter the following Script into the script window.

```
var wrd(50),pg(3);
input "Enter Search
Keyword:",wrd;

set pg = seq;
find trunc(wrd);

if pg = seq
{ beep;
 msg trunc(wrd):" not
found.","Click Mouse to
Continue.";
}

stop;
```

"Add Records" Button

This button is used to add new pages to the folder. It provides the user with an option of adding a page at the end of the folder or immediately after the current page. The button uses the `menu` statement to provide the user with a choice, then uses the `clone` statement to make a copy of a data page.

Each of the fields on the new data page are erased, then the user is prompted field-by-field to fill in the data.

This script uses the `object` statement which, in this case, allows several different fields to be referred to by a single statement within a loop. The loops in the script labeled `@LOOP` and `@LOOP2` count through the fields "S1" through "S10" and "T1" through "T10," first clearing the fields of their old contents, then prompting the user to enter new values. By using the concatenation operator (a colon) the loop counter (`i`) is appended to either the letter S or T to refer to each field in turn.

The key to the implementation of the `object` statement is naming the fields with names that create a numeric series: S1, S2, S3, etc. Thus the index counter can indicate each field used by the script.

```
var x(2),i(2);
menu x,22,
"Add After Last Page   ":
"Insert After This Page":
"Cancel                ";
if x = 1
{ noshow;go count;clone;show;}
if x = 2 clone;
if x = 3 stop;
```

```
set title="";
set artist = "";
set i = 0;
@loop
   set i = i + 1;
   object "S":i;
   set object = "";
   object "T":i;
   set object = "";
if i < 10 jump loop;
beep;
prompt "Enter CD Title";
do title;
prompt "Enter Artist";
do artist;
set i = 0
@loop2  set i = i + 1;
   object "S":i;
   prompt "Enter Song Title";
   do object;
   object "T":i;
   prompt "Enter Song Time";
   do object;
if i < 10 jump loop2;
stop;
```

"Delete Page" Button

The button to delete a page first checks to see if the current page is either the first page or the base page, and doesn't allow it to be deleted if it is either of these pages. After the sequence number is determined to be legal to delete, a pop-up menu is presented to the user that asks the user to confirm the delete before the script continues.

If the user answers "Yes," the script makes a copy of the page into a disc file called "undelete" and then deletes the page from the folder. Copying the page into a file allows for later retrieval of the page if desired, although none of the scripts in the folder are, at this point, set up to actually implement this retrieval.

```
var x(2);
if seq <=1
{ msg "Cannot Delete This
Page","Click Mouse To
Continue";  stop; }
menu x,22,
"Delete This Page?  No":
"                   Yes";
if x <= 1 stop;
if x = 2
 { cut "undelete";
   delete; }
stop;
```

The "Print" Button

This button uses the page printing command and is very simple. It prompts the user to prepare the printer, then prints a copy of the screen on the system printer. This will not work if the system printer is not capable of printing graphics.

```
msg "Prepare Printer,then",
      "Click the Mouse.";
cprint;
stop;
```

The "Autoexec" Button

LinkWay uses a special button on the base page of a folder called "Autoexec" to carry out certain initialization procedures whenever a folder is opened by the "Open" command of the Folder menu. The"Autoexec" button is usually a script button with statements to set up initial variable values, initialize serial ports, and other startup tasks. In this folder the "Autoexec" button is a simple one, turning off the menu bar and then going to the first page in the folder.

The "Autoexec" button can be located anywhere on the page, but it is usually a good idea to locate it in a consistent spot so that it can be located for editing, because it usually has no icon assigned to it and is invisible on the screen. As a convention, *LinkWay* scripters usually place it in the upper-left corner of the screen, just below the *LinkWay* Folder menu.

```
nombar;
go 1;
stop;
```

This folder idea can be adapted to a number of different applications, and the script buttons used with modification in other folders. The function of this folder could be changed by changing the labels on the base page to indicate a different type of data for the data pages and changing the heading on the base page. Field oriented data applications are usually quite similar to this and the buttons are designed to illustrate some universal techniques.

Simple folders such as this can be the basis for elaborate data management systems which link folder to folder and include other applications. *LinkWay* lets you easily start with an idea and build on it.

This New Paint Program Provides New Power for
HyperCard *Animation*

Studio/1
Electronic Arts

by HLM Staff

So who needs another black and white paint program? If you're a *HyperCard* user who wants to add first rate animation to your stacks,you do! And if that is not enough, *Studio/1* from Electronic Arts is an application that brings great graphic tools normally associated with high-end illustration programs to a bit-map paint program. Figure 1 shows the on-line help screen which provides an overview of the tools available.

The air-brush can be edited for size and flow. The Curve and Bezier features, although not as complete or mathematically oriented, are reminiscent of Adobe *Illustrator*. There is even a gradients feature that makes it easy to create 3-D shapes with directional lighting effects with shadowing.

This program is almost completely bit-mapped in its orientation and generally sticks to simple 72 dot-per-inch graphics. The latest object created, however, can be edited in much the same way you would alter it in an object-oriented environment like *MacDraw* or the object layer of *SuperPaint*. In addition, there are two different types of text available: bit-mapped, paint text and a "Text Layer" where PostScript text can be entered and edited. Thus if you are printing on high-resolution printers, like a LaserWriter, you will not have jagged text with any PostScript font you have available.

These excellent paint and draw abilities, plus many others too numerous to detail here, may not by themselves cause you to run out and plop down $150 (suggested retail). But its animation features set this product apart from every other paint program in this price range. We found that creating fast-moving, high quality animation sequences was easy and intuitive. And with *Studio/1's* license-free XCMD, incorporating animation into stacks has never been easier. The program saves animation in its own S1AN format for importing to *HyperCard,* but they can also be saved in PICS format for easy importing into many other programs including *SuperCard* and MacroMind's *Director*.

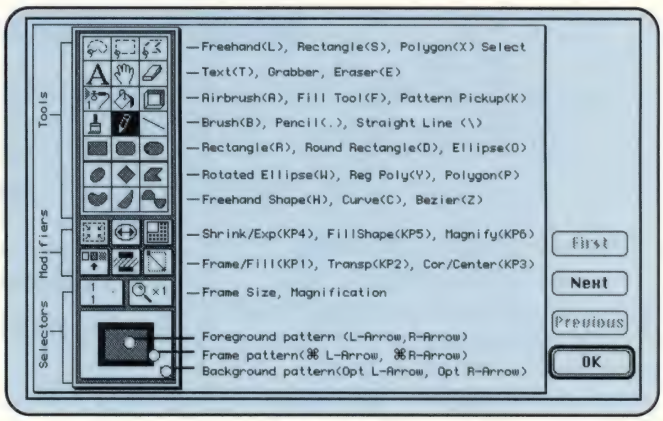


Figure 1

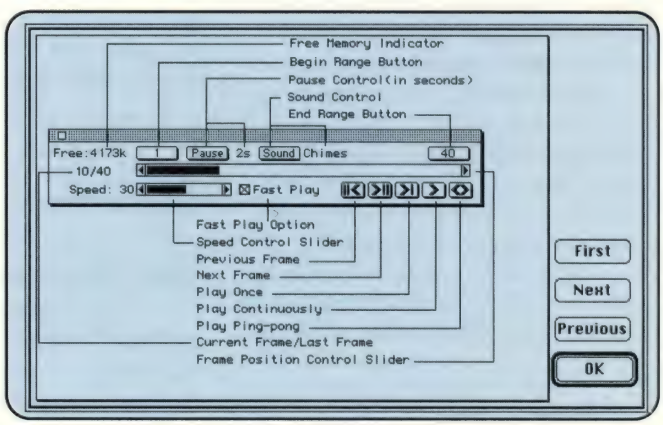


Figure 2

An Example

In *HyperLink*, we like to do more than just tell you that something is easy to do—we like to give you real examples so you can get the feel of how something is done. With this in mind, we modified the title screen of our introduction stack that comes on our *StackProjects* disk. Let's go step-by-step through the process so you can get an idea of how *Studio/1* works.

Our goal was to make the *StackProjects* logo zoom out from the center of the screen when the stack opens. We started *Studio/1* and selected "Set # Of Frames" from the Anim menu and entered 20 as the number of frames. The animation control palette (see Figure 2) was then displayed at the bottom of the screen (see Figure 3). This allowed us to scroll through the frames or play any part of the animation. The "Current Frame/Last Frame" indicator initially read 1/20. We then went to the last frame, frame 20, and created the graphic

Figure 3

This is the initial drawing from which we created our animation for the StackProjects *stack animation.*

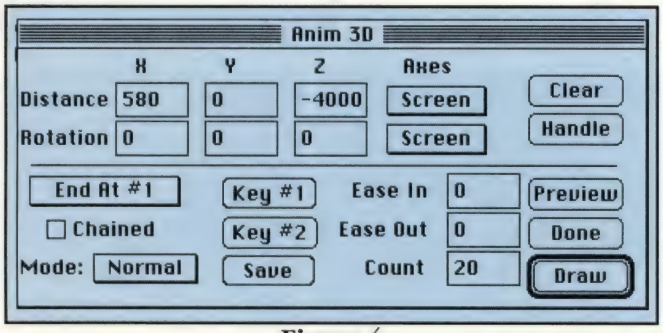


Figure 4

By entering the setting shown in this "Anim 3D" dialog box, we quickly created the animation effect of having our logo in Figure 3 zoom from a miniature version to full size.

shown in Figure 3. The text was created using *Studio/1's* Text tool, distorting it with the Quick Perspective tool, cleaning up with Fat Bits, and then adding the radiating lines below the text. Then we selected the entire graphic and chose the "Anim 3D..." option from the Anim menu bringing up the dialog box in Figure 4.

On the left side of this dialog box is a two option pop-up menu: "Start at #1" or "End At #1." We chose the latter because we wanted our animation to end with the graphic we had created. Clicking on "Preview" displays a box outline of how the animation will be drawn. We experimented with the values in the "Anim 3D..." dialog and found the data shown next to "Distance"—X=580, Y=0, Z=-4000—gave us the effect we desired.

When we were satisfied with our animation we clicked on the "Draw" button. This returned us to the main screen and each frame of the graphic was painted from 1 to 20. Now clicking on the "Play Once" button in the animation control palette played the animation.

To top off our animation, we put a little twinkle into the dot above the "j" in the word "Projects." To do this, we added 10 more frames by selecting "Set # Of Frames" from the Anim menu, and changed the number of frames from 20 to 30. The "Duplicate Last Frame" option was also selected so the graphic we had created would remain on the screen through frame 30.

Next we went to frame 29 and created an eight

Figure 5

We added 10 frames of animation and created a rotating "twinkle" on the dot of the "j" by entering the proper values into the "Rotation" boxes in the "Anim 3D" dialog box.

pointed "twinkle" (see the graphic overlaid on the dot above the "j" in Figure 5). We created this graphic off to the side so we could select it and drag it on top of the "j," leaving it selected. Next, Selecting "Anim 3D..." from the Anim menu we entered a rotation setting of Z=90 and a Count=9, and again chose "End At # 1." Clicking on the "Preview" button gave us a good idea of what would happen to our new graphic—it stayed in one place and rotated 90 degrees around its center. We then clicked on the "Draw" button to complete the animation process. Then we saved the file in *Studio/1's* own S1AN animation format as "Logo" to allow us to import and play the animation in *HyperCard.*

Importing to HyperCard

We used the *Studio/1* "Install" stack to place "Studio1" XCMD into the "StackProjects" stack. Then we added the following routine to our stack script :

```
on openStack
  go cd 1
  show btn "Cover"
  hide menubar
  hide msg
  put the rect of btn "Cover" into Spc
  Studio1 open,"LogoRef","Logo"
  if the result is empty then
    Studio1 Goto,LogoRef,Spc,Destination1
    Studio1 Forth,LogoRef,Spc,Destination29,¬
     Speed60
  end if
  hide btn "Cover"
  Studio1 close,LogoRef
end openStack
```

In addition to this script we added an opaque button called "Cover" which exactly covered the area where we wanted to display the animation. The rectangle coordinates of this button are placed into the variable `Spc`, and these coordinates are used to specify where the animation appears. The "Logo" animation file is then opened and the reference to that animation established in the variable `LogoRef`. This variable is used later to tell the XCMD which animation we are using. The `goto` command establishes the first frame as the starting place. The `Forth` command then instructs the XCMD to play the next 29 frames of the animation.

When completed, closing the animation file is essential, otherwise we found that memory limitations can cause *HyperCard* to freeze up, requiring a power down. But with this one caveat noted, we found the process both easy and fun.

Save $5.00 or More by Pre-registering Now

AppleFest'89 SanFrancisco

Discover How to Make the Most of Your Apple® Computer!

September 22-24, 1989
Brooks Hall / Civic Auditorium
San Francisco, California

APPLEFEST

It's the ultimate experience for Apple users — and prospective users. The all-new AppleFest®'89, at San Francisco's Brooks Hall and Civic Auditorium, September 22-24, 1989. Bigger and more exciting than ever, with more than 200 companies exhibiting the latest products for Apple II and Macintosh® computers.

There's more. Aside from all of the good things AppleFest '89 will do for your mind, registering right now will do something great for your wallet. You can save $5, $30 . . . even more, off of the regular admission price if you register now. This is your best chance to get the scoop on the hottest new Apple II and Macintosh products and applications. Like AppleWorks GS. HyperMedia. Desktop publishing and presentations. Using Apple computers in multi-media. And much, much more.

AppleFest '89. There's only one thing smarter than deciding to be there.

Deciding now.

Call 1-800-262-FEST to order tickets or for more information. (In Massachusetts, call 617-860-7100.) Remember: act today, to receive the advance registration savings.

Shafer On Scripting

Scripting Languages Outside HyperTalk

by Dan Shafer

In the Beginning, there was only HyperTalk. It was a prototypical and seminal piece of work. Its impact and influence have been so widespread it has spawned imitators, inspired other scripting language syntaxes, and even modified some of the most interesting aspects of the recently announced System 7.0 software from Apple Computer. Apple has spoken of a new approach to the way people use Macintoshes, an approach built around something called AppleScript. It isn't hard to figure out that this new "thing" is quite likely to bear a striking resemblance to its close ancestor, HyperTalk.

So now we are entering a new phase of a new era. Scripting has moved beyond *HyperCard* and HyperTalk. As *HyperLink Magazine* is committed to covering these other tools and languages, so this column should keep pace. I will be spending time in future columns looking at the scripting languages inherent in other products besides *HyperCard*. Not that I intend to abandon my HyperTalk interest. If anything, the emergence of these other languages and environments has resulted in a heightened respect for *HyperCard*, HyperTalk, and the whole concept. Bill Atkinson, Dan Winkler, *et al*, can lay legitimate claim to having spawned an entire new industry. Beyond software and into scriptware.

Someday, hopefully, all of these scripting languages will talk to one another. I shared such a vision with two technical gurus from Microsoft and Ashton-Tate at the MacWorld show in San Francisco earlier this year. That was the show at which Apple Veep Jean-Louis Gassée, enthusing over the *SuperCard* introduction, announced that Apple would help establish a sort of scripting language standards committee. Several people I talked with saw this as a harbinger of a Brave New World of End-User Programming. It may yet prove to be just that, in which case the revolution started by *HyperCard* may turn out to have as much impact on people's lives as computers have. Perhaps more.

Dan Shafer is the President of Strategy Consulting and author of many microcomputer books, including HyperTalk Programming (including version 1.2) and Understanding HyperTalk *(both from Hayden Books).*

What's a Scripting Language?

Let's start with at least a stab at a definition of a scripting language. Undoubtedly, this definition will evolve as innovators bring new ideas to the table. But for the moment, I suggest we start our discussion of the subject by defining a scripting language as a general-purpose computer programming language designed to deal with events and messages and at least partly accessible to people with little or no programming background.

Someday, hopefully, all of these scripting languages will talk to one another.

There are three components to this suggested definition. First, a scripting language must be a computer programming language. In other words, it must be able to make the computer do something. The "something" that it can make the computer do should not be so narrowly defined that it eliminates large numbers of end-user programmers from being interested in it. A language that deals, for example, with musical notation, may be accessible to people who are not programmers but its purpose is too narrow to qualify as a scripting language under this proposal.

Second, a scripting language must be designed to deal with events and messages. Why? I submit that there are several good reasons for taking this position. End-user programming will almost always involve events because end users see their interaction with the system in terms of events happening, not in terms of data being processed or algos being rhythmed (whatever that is, they exclaim). Modern microcomputer environments—Microsoft Windows, the Macintosh, X-Windows, etc.—are built largely around the event-processing model. Writing routines to respond to specific events in the system encourages the design and creation of small, compact, concise, single-purpose routines.

Finally, a scripting language must be at least partly accessible to people with little or no programming experience or background. It is probably clear enough that the language ought to be this approachable to qualify

for the epithet "Scripting Language." However, the word "partly" is also important; the language must have enough richness and depth that a programmer can use it, bend it, make it do things a less-experienced end-user programmer won't be comfortable doing. This may not be true in the future, but as long as systems are designed in such a way that the user is intimidated attempting to deal with any aspect of their behavior, this is a necessary component.

So there you have it, my first attempt at defining this new animal about which we are all becoming so excited. I welcome your comments, feedback, arguments, counter-definitions, or (could it be?) agreement.

The Field Today

There are probably more scripting languages lying around the dusty floor of computerdom than any of us individually realize. Here are the ones about which I know:

- HyperTalk (*HyperCard* - Apple Computer)
- SuperTalk (*SuperCard* - Silicon Beach Software)
- HyperScript (*Wingz* - Informix)
- modL (*Extend* - Imagine That! Software)
- PADTalk (*HyperPAD* - Brightbill-Roberts & Co Ltd.)

I'm sure there are others, and you're sure to tell me about the ones I've missed. But these are enough to start the discussion. [Another language that definitely fits Dan's definition here is the *PLUS* Programming Language (PPL)—Ed.]

Notice that all of the above scripting languages are Mac-based except PADTalk, which is the scripting language in the IBM PC *HyperCard* work-sort-of-like *HyperPAD*. Three of the four Mac languages are more or less well-known, the exception being modL. I thought for a few minutes before deciding this language fit my definition. It clearly fits the other criteria, but the issue of whether it is sufficiently general-purpose gave me pause. *Extend* (if you haven't seen it, you really ought to, assuming you're interested in the subject) is a simulation and modeling tool for the Mac. The question is: Is simulation and modeling a general purpose for a computer? I concluded that it is, based on the fact that you can simulate and model almost anything you can think of that happens in the real world.

Listing 1 - HyperTalk equivalent of Listing 3

```
on openStack
  go to last card
  if field "Description" is empty then
    select before text of field "Description"
  else
    send mouseUp to background button "New"
  end if
end openStack
```

Listing 2 - HyperTalk equivalent of Listing 4

```
on mouseUp
  set lockScreen to true
  go to last card
  get field "Index Number"
  set lockScreen to false
  doMenu "New Card"
  put it + 1 into field "Index Number"
  select before text of background field 2
end mouseUp
```

All of these languages have a similar, though not identical, core syntax that could be generally described as follows:

```
<event_word> <event_name>
[<optional parameters>]
  <body of handler>
<end clause>
```

In fact, the three popular Mac scripting languages [and the new PPL—Ed.] all use the same basic construct for event-related handlers:

```
on <event_name> [<optional
parameters>]
  <body of handler>
end <event_name>
```

(ModL is the exception; it uses a more C-like syntax, another factor which almost led me to conclude that it is not really a scripting language. But its syntax is otherwise fairly comprehensible.)

PADTalk takes a similar approach, but changes the keywords and adds some other requirements. The basic structure for a PADTalk script is:

```
handler <event_name>;
BEGIN
  <body of handler>;
END;
```

InterScript?

Ultimately, it seems to me, we have a chance here to do something with a computer programming language or environment that has eluded earlier efforts. We can make a truly accessible language available to all users of all computers. The key phrase is "all computers." If we can somehow get HyperTalk to talk to HyperScript and HyperScript to talk to modL, then maybe we can also get HyperTalk to talk to PADTalk. I am aware of one group that has already built a set of routines that take HyperTalk scripts and convert them to their PADTalk equivalents. But I am talking less about translation than I am about communication between scripts.

Wouldn't you like to be able, from HyperTalk, to call a HyperScript script in *Wingz*, have that script perform some complex calculation, and return its result to you in *HyperCard*, completely transparently to your user? Or to be able easily to use *HyperCard* as a front end for applications built in *Wingz* without having to resort to XCMD's and other deep, dark secrets of the Programming Club?

It seems to me that if we don't let this thing get too far down the road before we try to find ways of doing this kind of inter-script communication, we consultants can move the usability of our software up a notch or two in the eyes of the people who ultimately pay our salaries: the users.

So Where From Here?

As I read magazines and newspapers, hang out on several national and regional electronic bulletin

Listing 3 - PADTalk equivalent of Listing 1

```
HANDLER openPad;
BEGIN
  go to last page;
  if field "Description" is empty then
  send "Select" to field "Description"
else
send "Select" to button "New";
END;
```

Listing 4 - PADTalk equivalent of Listing 2

```
HANDLER Select;
BEGIN
  set lockScreen to true;
  go to last page;
  get field "Index Number";
  set lockScreen to false;
  doMenu "New Page";
  put it + 1 into field "Index Number";
END;
```

board systems, and talk with users and programmers all over the world, I sense the first stirrings of interest in scripting languages and their use. But the interest level in HyperTalk and *HyperCard* is, for the moment at least, eons ahead of the other scripting languages on all the other platforms combined. So it behooves me to keep my eye on the HyperTalk ball, with only occasional forays into other areas of similar interest.

At the same time, something phenomenal and interesting has happened: the level of HyperTalk scripting expertise "out there" has soared dramatically. This magazine is filled with scripts, some of which are quite elegant. Other more general-interest magazines such as *MacUser* and *MacWorld* also run full scripts (though you seldom if ever see a Pascal or C listing, for example; another indication of scripting's immense accessibility and resulting popularity). So the usefulness of a column like this one that provides scripting techniques is arguably near its end. More and more users are getting the point. They are saying to those of us who have considered ourselves to one degree or another pioneers, "Step aside, folks, and let us through."

Before you get your hopes up, though, I'm not stepping aside. Nope. I'm too stubborn for that. So instead, I'm repositioning. (That's what marketing people do with obsolete products and strategies, isn't it?) Future "Shafer On Scripting" columns will focus on the people in this community, the strategies of the companies involved, the broader issues of end-user programming, and similar matters. I'll toss in scripting ideas where they occur to me; sometimes they won't be HyperTalk but one of the other scripting languages.

It's up to you to help me steer this thing, though. Let me know how I'm doing in terms of covering the subjects you're interested in seeing. I'll try to be responsive.We are entering an incredibly exciting new era here; if we help each other get to the other end, the results may yet prove as staggering as we all hope.

HyperTalk to PADTalk

You didn't think I was going to let you off without a single script did you? Wrong!

I thought you might find it mildly interesting to look at two scripts in my personal filing stack as they appear in HyperTalk and as they look when converted to PADTalk, the scripting language used by *HyperPAD*. It's not that the scripts themselves are all that exciting; they're the kind of routine stuff all of us stackheads have to do and use all the time. But I thought it might be instructive to see the same functionality in both programs, particularly given that they are on different hardware.

Listing 1 is the HyperTalk listing of the openStack handler for my file stack. Listing 2 is the HyperTalk listing of the on mouseUp handler for the button in that stack that creates a new file index card when needed or on demand. Listing 3 is the same handler as Listing 1, only in PADTalk, while Listing 4 is the PADTalk version of Listing 2. If you think, after looking at them, that there are precious few differences between HyperTalk and PADTalk, that's only because there are, for the most part. I'll take a closer look at what some of those differences are in a future column. Meanwhile, let me say that I opened *HyperPAD* prepared to be disappointed. I expected no visual effects. Wrong. In fact, *HyperPAD* has some incredible visual effects, several of which aren't in *HyperCard*. I expected interface problems, particularly during pad design and construction. Wrong. The interface is very much like *HyperCard's*; even the control-key equivalents are the same, as much as possible.

If you're a *HyperCard* developer and you have clients who continue to insist on using the DOS platform, you could do worse than to pick up a copy of *HyperPAD* and see what you might be able to conjure up in it.

Interfacing The Future

Creating Animation in HyperCard *and* SuperCard
Part 2 of 2

by Craig Ragland

In the last issue we covered four important new *SuperCard* animation techniques, as well as five techniques available using scripting in both *HyperCard* and *SuperCard*. This column considers 11 techniques available in both products, as well as seven techniques supported by *SuperCard*.

Changing the Rectangle Property

Scripts that set the rectangle property of a button or field in *HyperCard* change the object's size. *SuperCard* also lets you set rectangles of paintings (which are cropped instead of scaled) and of draw graphics (which scale rather inaccurately). This script segment causes a button to stretch down and across the screen:

```
repeat with n = 1 to 20
  set rect of cd btn id 1 to ¬
 10,10,200+n*10,200+n*10
end repeat
```

Toggling Object Visible Properties

Scripts can set the visible property of a button or field in *HyperCard*. In *SuperCard*, both bit-mapped and draw graphics also have a controllable visible property. Scripts set the visible property of an object either using the set command (e.g., `set visible of cd btn id 1 to false`) or using the show and hide commands (e.g., `hide cd btn id 1`). This technique is particularly effective when used in combination with setting the location of an object or changing other properties (such as ICONs or Characters in a field with specialized fonts).

Automatic Drawing

Scripts can select different graphic tools, then use them to draw right on the screen in front of the user. This can have tremendous impact, as it seems sort of magical. Creative scripters have used *HyperCard* to create maps, draw business graphics (both 2 and 3-D!), write out words with calligraphy, add features to existing art, draw musical notes on a staff, and many other fancy drawings. *SuperCard* offers many new graphic tools which can be controlled by your scripts, though a few of the neater *HyperCard* paint tools and properties are missing (e.g., Lighten, MultiSpace).

Craig Ragland is a principal with INTERACTIVE DESIGN and co-author of The SuperCardHandbook, *published by Microsoft Press. He can be reached at 206-542-9000. INTERACTIVE DESIGN created* 101 Scripts & Buttons for HyperCard.

Scripts can select different graphic tools, then use them to draw right on the screen in front of the user. This can have tremendous impact, as it seems sort of magical.

Dragging Selected Paint Regions

In *HyperCard*, paint cannot be manipulated as independent objects. A method of animating paint (which animates a car and beachball in the Apple *HyperCard* sample stacks) is to select the region by dragging the selection tool, then `drag` the selected region around on the screen. *SuperCard* lets you place paint into a bitbox and then use the *SuperCard* `move` command to animate it as an object instead of a selected region.

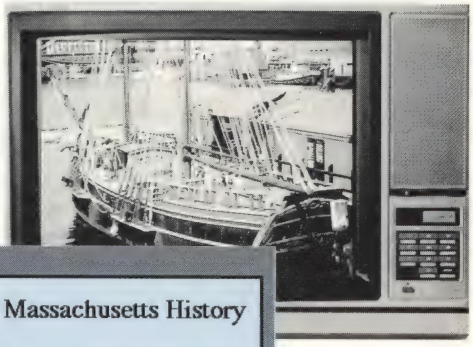
Dragging Selected Objects

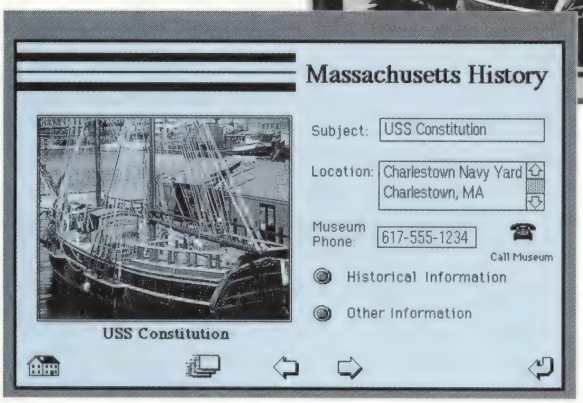
In *HyperCard*, a button or field can be selected using the button or field tool and then dragged around on the screen using a series of `drag` commands. This method can be used for graphics as well in *SuperCard*, but this method is less powerful and flexible than using the *SuperCard move* command to animate objects. Here is a *HyperCard* script segment that illustrates this technique (selecting a button automatically chooses the button tool):

```
set dragSpeed to 100
select cd btn id 1
drag from loc of cd btn id 1 to 100,100
```

INTRODUCING HYPERVISION™

for the ProViz™ Video Digitizer. The first Real–Time video software designed exclusively for HyperCard.™





Real–world images increase the educational power of HyperCard.

Video to HyperCard...

Now you can capture virtually any video image *directly* into your HyperCard stack, without the need for still models or a freeze-frame. With just one click of the mouse you'll instantly grab images from camcorders, VCRs, laser disk players—even television.

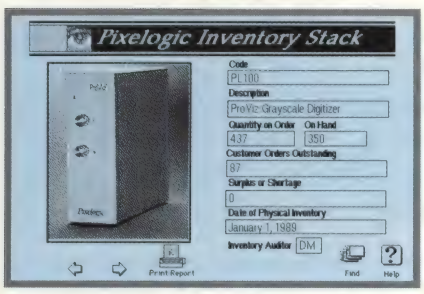
HyperVision is the new software interface to HyperCard for the line of ProViz Video Digitizers. Because HyperVision was developed specifically for the ProViz Digitizer and incorporates Apple's HyperScan™ software, you can capture and display images with unparalleled clarity.

Real-time and more...

- Real-time image capture for frame-grabbing or live pictures.
- Image scaling and cropping.
- The dithering algorithms from Apple's HyperScan software.
- The extra features of the ProViz software, including image capture in 16 shades of gray and 256 shades of gray.

Real–time images bring personnel stacks to life.

Real images bring product information into focus.



Just imagine what you can do with real-world images in your stacks:

- Add frames to the HyperCard front-end of an interactive videodisc.
- Supply your customers with an electronic sales catalog.
- Add product shots to your on-line documentation and demo disks.
- Incorporate photos into your electronic mail, personnel, or security system.

...on the entire Macintosh family.

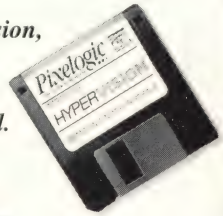
All you need to run HyperVision and the ProViz is:

- Any Macintosh—a Plus, SE, SE/30, II, IIx, or IIcx—with 2 Mb of RAM and a hard disk.
- System version 6.0 or later
- HyperCard version 1.2 or later.

Free Demo Disk

Our self-running demo disk will show you the simplicity and power of **HyperVision**. Call our toll-free number today and see for yourself why we say:

With HyperVision, HyperCard never looked so good.



1•800•432•2292

(in Massachusetts 617•938•7711)

Pixelogic℠

The Vision of the Future

Changing Button or Field Styles

Changing a button or field's style under script control can draw attention to that object. Here's an example that causes a field's border to flash:

```
repeat until the mouse="down"
  set the style of card ¬
  field  id 1 to rectangle
  set the style of card ¬
  field id 1 to opaque
end repeat
```

Changing Field Scrolls

Changing a field's scroll property under script control can move the contents of that field through the user's viewing area. *SuperCard* uses this technique within an `idle` handler to scroll credits in its Runtime Editor's *"About SuperCard"* dialog box. Here is an example script segment that illustrates this technique:

```
repeat with n = 1 to 10
  set the scroll of cd fld ¬
  id 1 to 10*n
end repeat
```

Selecting Objects

Selecting a button or field in *HyperCard* causes its border to turn into moving ants. Selecting an object in *SuperCard* causes its grow handles to be shown. These very different visual effects can be used in creative ways. In *SuperCard*, toggling the selected property of a small object on and off looks a bit like a twinkling star. *HyperCard* does not allow direct access to the selected property so scripts will have to directly select the button or field. Furthermore, the *HyperCard* moving ants effect is only seen if the script ends with an object selected.

Selecting Text Spans

Mac users know that selected text has special meaning. Selected text is inverted and appears white-on-black instead of black-on-white. By repeatedly selecting and unselecting a particular span of text you can be pretty sure people will look at it:

```
repeat until the mouse="Down"
  select line 2 of cd fld id 1
  wait 10
  select empty
end repeat
```

You can also select all the text in a particular field: just replace the words `line 2` in the above script with the word `text`.

Changing Field Text Properties

HyperCard scripts can change the various text properties for a whole field. These text properties include style, alignment, size, font, and height. In *SuperCard*, the style, size, and font properties can also be modified for a particular span of text within a field. Modifying these properties in either environment can create very dramatic text-based animation effects.

Changing Field Lines & Margins

Both *HyperCard* and *SuperCard* allow a field's showLines and wideMargins properties to be toggled from true to false. However,

HyperCard's showLines property has no visible effect on scrolling fields.

SuperCard Specific

Changing Shadow Offsets and Patterns

SuperCard buttons and fields have a shadow property that moves a drop shadow of an object by a specified number of pixels. This can be systematically modified to create an interesting animation where the drop shadow drops away or toward the button or field. For example:

```
repeat with n = 4 to 30
  set the shadow of card ¬
  button 1 to n
end repeat
```

An undocumented feature of *SuperCard* is that you can also modify the shadow's pattern under script control by setting the button or field's ink property. Valid shadow pattern settings are Shad1, Shad2, ... , Shad10. Here's a script segment that cycles the pattern of card button id 1:

```
set the shadow of card ¬
button 1 to 4
repeat with n = 1 to 10
  put "shad" & n into holder
  set the ink of cd btn ¬
  id 1 to holder
  wait 10
end repeat
```

Changing Inks

A *SuperCard* graphic's ink property affects the way it interacts with other graphics. As indicated above, the ink property for buttons and fields affects the object's shadow pattern. Changing the ink property of a graphic that overlaps other graphics under script control can be very dramatic.

Changing Graphic Shapes

Because *SuperCard* graphics can be object-oriented, as opposed to just bit maps as in *HyperCard*, and they have a well developed suite of appropriate properties, you can change a draw graphic's shape using a few different methods. Fundamentally, you can either change the style of a draw graphic or modify object-specific properties (e.g., the points property of a polygon, the startAngle property of an Arc, or the roundWidth property of a roundRect). Changing the style of a graphic from a rectangle to an oval, for example, simply replaces the rectangle object with an oval having the same rectangle property.

More interesting is the ability to redefine the points property of either polygon buttons or polygon draw graphics. Changing the points property of a polygon could be used to gradually modify one complex polygon into another. For example, a simple man's boot could gradually be converted into a map of Italy.

Cycling Patterns

SuperCard allows scripts to control both the fill pattern and the pen pattern for different objects. This is standard in advanced Macintosh graphic software, but controlling these properties under script control offers interesting animation effects. By cycling an object's fill pattern through progressively darker patterns it becomes increasingly dominant in a particular card's art.

Changing Border Sizes and Shapes

In *SuperCard*, the border of an object can be independently controlled. Properties that affect the border's size and shape are the penHeight, penWidth, and lineSize. Setting the lineSize property simultaneously sets the object's penHeight and penWidth to that same value. If either the penHeight or penWidth properties are modified, the lineSize becomes the truncated average of the penHeight and penWidth values.

Changing any of these three values systematically causes the border to swell or shrink. The visual affect depends on the shape and type of the object being changed, as well as the current penPat property (pattern used for the border). Changing the border of fields changes the layout of any text they may contain. Changing the borders of rectangles has a very different feel from changing the borders of an oval. Here is a script segment that resets the penWidth of a graphic:

```
repeat with n = 1 to 24
  set the penWidth of card ¬
  graphic 1 to n
end repeat
```

Changing Arcs

The graphic style Arc has two alterable properties, the startAngle and the arcAngle. The startAngle determines the degree position for the arc's start, although the arcAngle is the arc's span. Changing the startAngle causes the arc to turn around a centerpoint, while changing the arcAngle, causes it to increase or decrease its angle size. Modifying these properties for arcs allows rather wild effects, particularly if they are used in conjunction with other animation effects (e.g., changing the arc's rectangle, cycling its color or pattern, etc.). This script segment causes an arc to spin around:

```
repeat until the mouse is down
  repeat with n = 1 to 11
    set the startAngle of ¬
    card graphic 1 to n*30
  end repeat
end repeat
```

Setting Window Locations, Scrolls, and Sizes

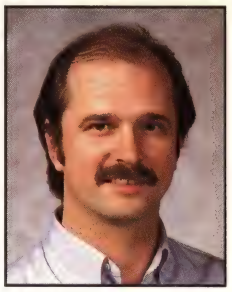
SuperCard scripts can set the location, scroll, and size of windows in addition to other *SuperCard* objects. This allows you to create multiple window animations by changing window properties. You could, for example, have a multiple window layout where different windows expand and display graphics based on the user's actions. By setting the scroll of a window, you could scroll a huge title across the screen. Unlike fields, which must be of the scrolling style to have a scroll property, all *SuperCard* windows have scroll properties which can be set under script control. Setting the scroll of windows that lack scroll bars offers great opportunities for very unexpected effects. Here's a script segment that might be used to scroll a window to the viewer's left (variations could also scroll it up or down or to the right):

```
open window "My Example"
repeat with n = 1 to 20
  set scroll of window ¬
  "My Example" to n*40,0
end repeat
```

In Conclusion

In this two-part column we covered 27 different animation techniques available using scripting in *HyperCard* and/or *SuperCard*. There are also many XCMD-based methods that third party developers have created, but these generally add higher costs and learning requirements to your projects. Animation adds greatly to the appeal of hypermedia and I hope you find these methods helpful.

A HyperCard, SuperCard, *and* IBM LinkWay
Application for Keeping Track of Your...



Baseball Card Collection

by Roger Wood

My father's basement is legendary. "One of these days" he always says, "I'll get the basement cleaned out." Somewhere in that basement is my childhood: stuffed animals, old baseball gloves, football helmets, games, and—my brother and I always say—our baseball card collection!

Just as we were putting together our latest issue my father came for a visit and, lo and behold, he'd found that collection and brought it with him. It dates back to 1952 and numbers in the thousands of cards, mostly from the 1950's. My father said, "Before we split these up between you and your brother, I'm going to sit down and catalog them and find out how much they're worth on the open market." Before I knew it he had purchased Dr. James Beckett's *Official 1990 Price Guide to Baseball Cards* (published by Ballentine books), and he was looking for a legal pad and a calculator. I said "Wait! Don't put it on paper, put it into *HyperCard,* and I'll print out all the lists you want and do all the calculating, too." Soon, I had my father in front of the computer and this issue's "Baseball Card Collection" StackProject was born.

This easy-to-understand-stack not only lets you catalog the cards, but you can scan in the pictures for easy reference. It also includes a sorting and totaling feature, so you can find out exactly how many cards you have, how many of each card of identical quality, and thus how much the total collection is worth.

Also, in keeping with *HyperLink's* increased coverage we have implemented the application not only in *HyperCard,* but *SuperCard* and *IBM LinkWay* as well. So that the maximum number of readers can use the *SuperCard* version, we have kept it simple—just create the *HyperCard* version, then convert it using *SuperEdit's* "Convert Stack" command from its File menu. For those with color Macs with larger memory, we have created an enhanced version of the project that uses custom menus and multiple windows. Color graphics can be scanned in and imported to the project via the Scrapbook desk accessory or the clipboard. This special *SuperCard* project is on this issue's *StackProjects* disk.

All of the versions explained here allow for the importing of black and white graphics using hand-held scanners. We will explain the process in detail along with what we found to be the best techniques for employing this inexpensive scanning method.

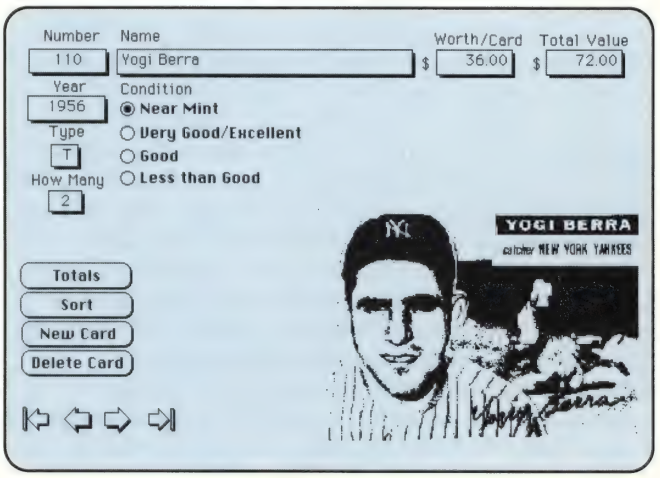


Figure 1
The HyperCard *"Baseball Card Collection" stack's main screen. All fields are of shadow style. The name of each field is directly above it in the background drawn with paint text.*

HyperCard and *SuperCard*

To get started, go to the last card in your "Home" stack and ensure that your User Level is set to "Scripting." Next choose "New Stack" from the File menu, name the new stack "Baseball Cards," and de-select the "Copy Current Background" check box so you will start with a blank screen. Go ahead and enter the script from Listing 1 into the stack script by choosing "Stack Info..." from the Objects menu, and clicking on the "Script" button.

Next go into the background of your new stack (type ⌘-B) and create the card shown in Figure 1. Before you create anything on this card, name the background "The Cards" be selecting "Background Info..." from the Objects menu and entering the Name in the dialog box that appears. Click "OK" once you've named the background.

Now tear off the Tools palette, and select the Text tool (the large A in the lower-left corner of the palette). Click on the proper parts of the screen and type in

Roger Wood is the Editor of HyperLink Magazine

"Number," "Name," "Worth/Card," "Total Value," "Year," "Type," "How Many," and "Condition." While still in the background select "New Field" from the Objects menu. When the field appears double-click on it and the "field Info..." dialog box appears. Select "shadowed" as the style of the field, and name it "Number." Now select "Text Style..." from the Edit menu (or press ⌘-T) and select 12 point, Geneva, centered. Then grab the lower-right corner of the field and resize it to be one line of text high and leave enough room for approximately 4 characters in width as shown in Figure 1. Next move the field up under the paint text which says "Number."

Now "clone" this field to create the other fields on the screen. You do this by holding down the Option key while clicking and dragging on the "Number" field you just created. Place the next field under the paint text saying "Year," then double-click on the field to bring up its "Field Info" dialog box, and name this new field "Year." You are going to be putting scripts in these fields in a minute, but first get them created, named, sized, and in place. Notice that the "Type" and "How Many" fields both contain centered text just like the "Number" and "Year" fields. The "Name" field should be set up to be aligned left, and the "Worth/Card" and "Total Value" fields should both be aligned right.

You need to create one more background field, but it is different from the others and is used to store some information for later report printing. The name of the field is "Condition" and it is invisible (which explains why you don't see it in Figure 1). After you create and name this field, you need to type "set the visible of field Condition to false" into the Message box. This newly created field disappears, but you will make use of it later.

Now you are ready to create the eight background buttons at the lower-left part of of the screen. Four of the buttons are rounded rectangle style: "Totals," "Sort," "New Card," and "Delete Card." Create one button by selecting "New Button" from the Objects menu. Double-click on this button to bring up the "Button Info..." dialog box and name the button "Totals." Now clone the other three rounded rectangle buttons by clicking and dragging on this button with the Option key down just as you did with the fields above, naming each one as shown in Figure 1.

The other four background buttons are the navigation buttons at the lower-left corner of the screen. The quickest way to create these is to select the button tool from the Tools palette, and then click and drag while holding down the ⌘ key. This creates a transparent button whose name is not showing. Create one button this way and double-click on it to bring up its "Button Info..." dialog box. Now click on the "Icon" button in this dialog box and all of the standard ICONs that come with *HyperCard* should show up in a scrolling field in an ICON dialog. Select the ICON for one of the buttons shown in Figure 1. As you create these navigation buttons put the scripts in each one. The scripts for the buttons, from left to right are as follows:

```
on mouseUp
  visual dissolve
  go first cd of this bg
end mouseUp
```

Listing 1 - Stack Script for the "Baseball Card Collection" stack

```
on openStack
  global start
  put true into start
  go card 1 of bg "Totals"
end openStack

on doMenu which
  if which is "New Card"
  then
    lock Screen
    doMenu "Copy Card"
    doMenu "Paste Card"
    set the name of btn 5 to
    "Radio Control 0,0"
    repeat with i = 1 to 4
      set the hilite of btn i to false
    end repeat
    repeat with i= 1 to the number of ¬
    bg flds
      put empty into fld i
    end repeat
    unlock screen with visual scroll left
    TabKey
  else if which is "Delete Card" then
    answer ¬
    "Do you want to delete this card?"¬
    with "Cancel" or "Yes"
    If it is "Cancel" then exit doMenu
    else send "doMenu Delete Card" to ¬
    HyperCard
  else pass doMenu
end doMenu

function CheckNum Num
  put 0 into decimals
  repeat with curnum = 1 to length(Num)
    set cursor to busy
    if char curnum of num = "." then add ¬
    1 to decimals
    if char curnum of num is in ¬
     "1234567890." and ¬
    decimals < 2 then next repeat
    else
      beep
      return "Error"
      exit CheckNum
    end if
  end repeat
  if num = "." then
    beep
    return "Error"
    exit CheckNum
  end if
  return Num
end CheckNum
```

```
on mouseUp
  visual dissolve
  go previous cd of this bg
end mouseUp

on mouseUp
  visual dissolve
  go next cd of this bg
end mouseUp

on mouseUp
  visual dissolve
```

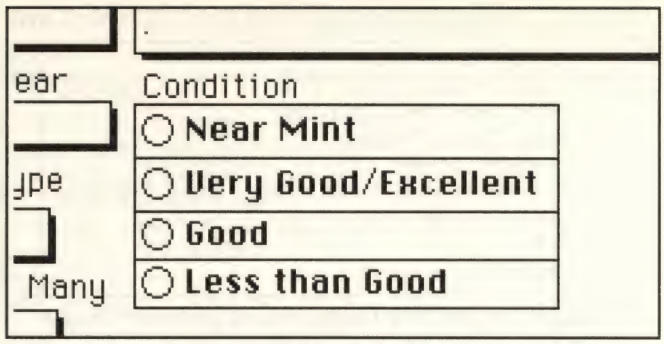


Figure 2

Place the radio buttons which indicate the condition of the baseball cards adjacent to one another. This way when you cover the radio buttons with the "Radio Control" button they will all be underneath it with no spaces in between them.

Listing 2 - Script for card button "Radio Control 0,0"

```
on mouseUp
  put the clickLoc into thisClick
  hide me
  click at (last word of the short name ¬
  of me)
  click at thisClick
  get the short name of me
  put thisClick into last word of it
  set name of me to it
  show me
end mouseUp

on newButton
  get the short name of me
  put "0,0" into last word of it
  set name of me to it
end newButton
```

Listing 3 - Script for background field "Number"

```
on closeField
  set numberformat to "####"
  put checkNum(me) into me
  if me ≠ "Error" then add zero to me
end closeField
```

```
 go last cd of this bg
end mouseUp
```

Clone the other three navigation buttons as you did before, open each button's info dialog box, and select the appropriate ICON as shown, then open each one's script window and enter the appropriate script from above. There are a couple of shortcuts for editing the script of a button. Either you can double click on the button while holding down the shift key (with the Button tool selected), or single click on the button while holding down both the Option and ⌘ keys. This second shortcut works whether the Button tool is selected or not. You have now created all the fields and buttons on the background layer of this card.

The "Radio Control" Button

Now create the radio buttons for indicating the condition of the baseball cards. We use a special technique submitted to us by Doug Weathers called the "Radio Control Button" to take care of this group of buttons. The details of this method are explained in this issue's "HyperLink Tips" column.

To start creating these buttons, first get back to the card level of the card by pressing ⌘-B, and choose the Button tool. Create a new button, and open its "Button Info..." dialog box by double-clicking on it. Select radio button as the style and type in "Very Good/Excellent" for its name because this is the longest radio button name. Also make sure that the "Show Name" and "Auto Hilite" check boxes are selected so they can be easily identified and will operate properly. Because all of these radio buttons have the same script, enter it into this one so it will be in all of the radio buttons that you create by cloning this one. The script for this button is:

```
on mouseUp
  put the short name of me into fld condition
end mouseUp
```

Now clone the other radio buttons as shown in Figure 2, naming each as you go along. Place the four radio style buttons so they are just touching and form a perfect rectangle. This is so the special "Radio Control" button will exactly cover all of them. Now create a transparent button. Do this by selecting the Button tool, hold down the ⌘ key, and click and drag. Open this button's "Button Info" dialog box, and name it "Radio Control 0,0." Open its script window and place the script in Listing 2 in it. Now when you click on any of the radio buttons in the group, the button you click on is selected and all others in the group are automatically turned off. Please be aware that the name of the "Radio Control" button itself is altered to keep track of which radio button is currently selected.

Object Name	Listing Number
"Radio Control 0,0" Button	2
"Number" Field	3
"Year" Field	4
"Type" Field	5
"How Many" Field	6
"Name" Field	None
"Worth/Card" Field	7
"Total Value" Field	None
"Totals" Button	8
"Sort" Button	9
"New Card" Button	10
"Delete Card" Button	11

Figure 3

This is a guide to identifying which Listings in this article go to which fields and buttons on the screen shown in Figure 1.

The Scripts for "The Cards"

You are now ready to put the rest of the scripts into the objects. Figure 3 contains a list of all of the scripts of the objects used on the card of Figure 1 and the Listing number for each. The stack script is the longest and has special handlers for creating and deleting cards, along with a numeric checking handler called `checkNum`. Enter these scripts and you are ready to create a new background called "Totals."

Listing 4 - Script for background field "Year"

```
on closeField
  set numberformat to "####"
  put checkNum(me) into me
  if me ≠ "Error" then add zero to me
end closeField
```

Listing 5 - Script for background field "Type"

```
on closeField
  if length of me ≠1 then put "?" into me
end closeField
```

Listing 6 - Script for background field "How Many"

```
on closeField
  set numberformat to "#"
  put checkNum(me) into me
  if me ≠ "Error"
  then
    add zero to me
    set numberFormat to "#.00"
    put (me) * fld "Worth/Card" into ¬
    field "Total Value"
  end if
end closeField
```

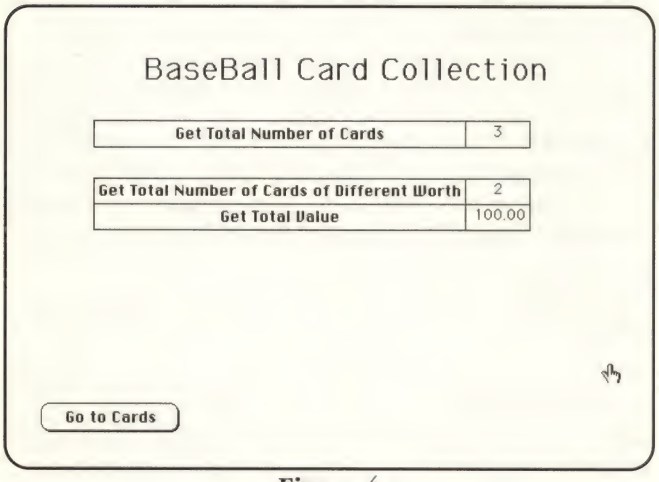


Figure 4

The "Totals" Card acts as a title screen and a focal point for all of the stack's calculations.

The "Totals" Background

So the stack can calculate the total number of cards and how much all the cards cataloged in the stack are worth, create the "Totals" card (see Figure 4). Begin by selecting "New Background" from the Objects menu. A blank screen appears. Select "Background Info..." from the Objects menu, enter the name "Totals," then enter the script for the background by clicking on the "Script" button and keying in Listing 12.

Everything on the "Totals" card is at the card level, so get out of the background level by pressing ⌘-B. Now place the title "Baseball Card Collection" at the top. You can either create a field or use paint text. Now select "New Button" from the Objects menu, and open it by double-clicking on it. Name it "Go to Cards." Now click the "Script" button and enter Listing 13.

Next create three fields to hold the values of the totals that are calculated by the three calculation but-

Listing 7 - Script for background field "Worth/Card"

```
on closeField
  set numberformat to "#.00"
  put checkNum(me) into me
  if me ≠ "Error" then
    add zero to me
    put (me) * fld "How Many" into field ¬
    "Total Value"
  end if
end closeField
```

Listing 8 - Script for background button "Totals"

```
on mouseUp
  go to bg "Totals"
end mouseUp
```

Listing 9 - Script for background button "Sort"

```
on mouseUp
  answer "By What?" with "Worth/Card" ¬
  or "Year" or "Last Name"
  if it is empty then exit mouseUp
  set cursor to watch
  lock screen
  if it is "Last Name"
  then sort by last word of field Name
  else if it is "Year"
  then sort numeric by field "Year"
  else if it is "Worth/Card"
  then sort descending numeric by ¬
  field "Worth/Card"
  go first card of bg "The Cards"
  unlock screen
end mouseUp
```

Listing 10 - Script for background button "New Card"

```
on mouseUp
  doMenu "New Card"
end mouseUp
```

Listing 11 - Script for background button "Delete Card"

```
on mouseUp
  doMenu "Delete Card"
end mouseUp
```

Listing 12 - Script for background "Totals"

```
on doMenu which
  if which = "New Card"
  then answer "No New Totals Cards" ¬
  with "OK"
  else if which = "Delete Card"
  then answer "Can't delete Totals Card" ¬
  with "OK"
  else pass doMenu
end doMenu
```

Listing 13 - Script for button "Go to Cards"

```
on mouseUp
  global start
  if start is true
  then
    put false into start
    go to first cd of bg "The Cards"
  else go back
end mouseUp
```

Listing 14 - Script for button "Get Total Number of Cards"

```
on closeField
  set numberformat to "#.00"
  put checkNum(me) into me
  if me ≠ "Error" then
    add zero to me
    put (me) * fld "How Many" into field ¬
    "Total Value"
  end if
end closeField
```

Listing 15 - Script for button "Get Total Number of Cards of Different Worth"

```
on mouseUp
  set cursor to busy
  put the number of cards in bg ¬
  "The Cards" into cd fld ¬
  "Number of Different Cards"
end mouseUp
```

Listing 16 - Script for button "Get Total Value"

```
on mouseUp
  push card
  lock screen
  go bg "The Cards"
  put zero into totValue
  repeat with i = 1 to (the number of ¬
  cds in bg "The Cards")
    put (fld "Worth/Card") * (fld #) + ¬
    totValue into totValue
    set cursor to busy
    go next
  end repeat
  pop card
  set numberFormat to "#.00"
  add zero to totValue
  put totValue into cd fld "Total Value"
end mouseUp
```

tons. These fields are called, from top-to-bottom, "Number of Cards," "Number of Different Cards," and "Total Value." They are rectangle style, with a text style of 12 point Geneva, and their alignment is centered. You can create one and clone them as explained above.

Now create the buttons that calculate the values for these fields. They are also rectangle style and their names are, from top-to-bottom, "Get Total Number of Cards," "Get Total Number of Cards of Different Worth," and "Get Total Value. The scripts for these are in Listings 14, 15, and 16 respectively.

Now you have the basic stack ready for your baseball card collection. Needless to say, the structure could be used for myriad other stacks for keeping track of any collection—from coins, to stamps, to you name it.

Converting to *SuperCard*

This stack converts to *SuperCard* without any difficulty at all. Just open *SuperEdit* and choose "Convert Stack" from the File menu and select the "Baseball Card Collection" stack in the "Get File" dialog box. There is only one small change in the stack script that must be made, and it can be done after the conversion is complete. The change is in the stack script which becomes the window script in *SuperCard*. Just change the line:

```
else send "doMenu Delete Card" to HyperCard
```

to read:

```
else Delete Card
```

It is interesting that even though the new card and delete card functions are handled as menu options in *HyperCard* and commands in *SuperCard*, the `doMenu` handlers generally require no modification when converted to *SuperCard* as long as they are called from within buttons as the "New Card" and "Delete Card" buttons are in the "Baseball Card Collection" stack. Obviously the `send to HyperCard` message, however, does have to change because you can't `send` a message to *HyperCard* from within *SuperCard*.

The *LinkWay* Version

The *LinkWay* version of this application has nearly all of the same functions as the *HyperCard* and *SuperCard* versions. We left out the sort routine because *LinkWay* does not, in its present version, have a built-in sort function. The only other differences are strictly cosmetic. Because different backgrounds are not possible, we included the totaling function on the base page of the folder, and thus the totals appear on every page.

Because the scanning we are featuring in this application uses the Logitech hand-held scanner, and it only works in black and white, we chose to create our application in MCGA mono mode. This creates the high resolution screen shown in Figure 5.

If you wish to create this application in another graphic mode, the only changes will be in the exact placement of buttons and fields—all the scripts should function identically.

Start by typing "LINKWAY /H" at the DOS prompt and you are taken to the MCGA mono screen. Select "New" from the Folder menu, name your new application "BBALL," and you are taken to the base page. Start by setting up the objects shown in Figure 5—the base page of our BBALL folder.

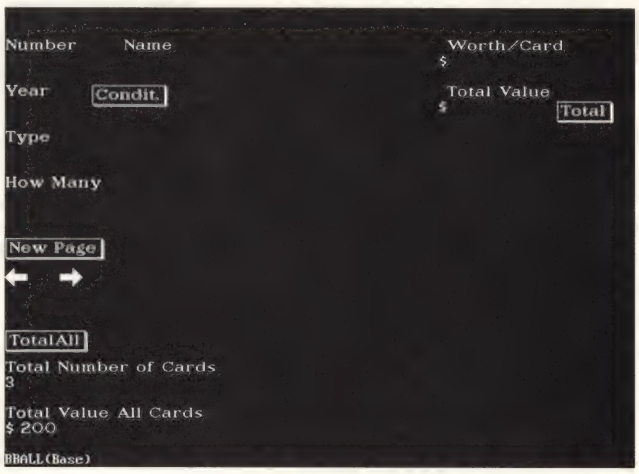


Figure 5
This is the base page of the IBM LinkWay *version of our baseball card collection folder called "BBALL."*

You should start by creating the label fields shown in Figure 5. These can be created one at a time and positioned. It is a good idea to put text in them right away, because empty fields can get lost very easily.

To create the label "Number" in the upper-left corner, for example, select the "New" option from the Object menu. A pop-up menu appears asking whether you want to create a button, picture, or field—choose "Field." Next, your mouse cursor changes to a small dotted box. Place the box in the upper-left corner of the screen (stay below the menu bar), and click once. Now move down and to the right to make the field large enough to hold the text you want to put into the label, in this case "Number" and click again. Now a small window comes up showing several samples of font sizes that you can use. Select the font that is the second from the top (12 point Roman).

Next, a pop-up menu appears that allows you to set the name of the field, its size (in character width and number of lines), and whether the field is locked or unlocked. These label fields should not be given any names as the text they contain differentiates them sufficiently. You can count the number of characters for each label and all fields are one line high. Leave each field unlocked and put the label name in each field as you create it. Don't forget to create small "$" fields to label the data where appropriate.

There are two fields in the lower-left corner of the base page that are used to hold the totals for the entire folder. One is the total number of cards in the baseball card collection, and the other is the total worth of the collection. These fields go directly below their labels and require a field name so we can have scripts place values into them. The upper field (just below the "Total Number of Cards" label) is named "TotCdfld" and the lower field (below the "Total Value All Cards" field is named "TotVfld." All of the other fields must appear on every data page, so they will be placed on page 1, rather than the base page. We will get to this page in a minute, but first let's finish setting up the base page with its buttons.

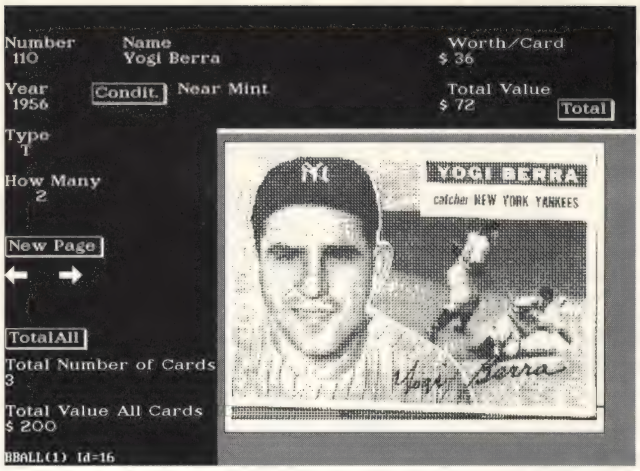


Figure 6

This is the data page of the folder called "BBALL." The "New Page" button uses the clone command, thus all fields, buttons, and pictures must be in place before the folder is ready for use.

The Buttons

There are four shadow type buttons on the base page: "New Page," "Total," "TotalAll," and "Condit." Create these by selecting "New" from the Object menu and choosing "Button" in the first pop-up menu. You are asked which of *LinkWay's* button types you wish—for each of these select "Script." You are then prompted for a button title, note that the names are a maximum of 8 characters. Then you are shown a number of button types. For each one select the standard shadowed button type with its name showing in each. Finally, you are asked to enter a script for the button. Here are the scripts for these four buttons:

"New Page" Button:

```
go count
clone

set numfld = blank;
set namefld = blank;
set yearfld = blank;
set typefld = blank;
set howmyfld = blank;
set condfld = blank;
set worthfld = blank;
set totfld = blank;
prompt "Card Number?";
do numfld;
prompt "Player Name?";
do namefld;
prompt "Year of Card?";
do yearfld;
prompt "Card Type?";
do typefld;
prompt "How Many?";
do howmyfld;
prompt "Card Worth?"
do worthfld;
```

"Total" Button:

```
var totVal(10);

set totVal = Worthfld*
Howmyfld;
set Totfld=totVal;
```

"TotalAll" Button:

```
var totAll(10),totVal
(10),Cdtot(10)
var x(3), curpage(3);

set Cdtot = 0;
set totAll = 0;
set x=0;
set curpage = seq;

@loop1
set x=x+1;
if x>count jump outofhere;
go x;
set Cdtot = Howmyfld + CdTot;
set totVal =
Worthfld*Howmyfld;
set Totfld=totVal;
set totAll=totVal + totAll;
jump loop1;

@outofhere
go 0;
set TotVfld = totAll;
set TotCdfld = Cdtot;
go curpage;
```

"Condit." Button:

```
var zx(1), Cond(19);

menu zx,19,
"Near Mint          ":
"Very Good/Excellent":
"Good               ":
"Less Than Good     ":
"Cancel             ";
if zx=0 | zx>=5 stop;

if zx=1 set cond = "Near Mint"
if zx=2 set cond = "Very
Good/Excellent";
if zx=3 set cond = "Good";
if zx=4 set cond = "Less Than
Good";
set condfld = cond;
```

Don't try to execute these button scripts right away. They require fields on both the base page and the data page, and until those fields are in place you will get error messages.

There is one other script button on the base page—an invisible one in the upper-left corner. This is a button called "Autoexec" that

appears on the base page of many *LinkWay* folders. It is used in much the same way that an `openStack` script is in a *HyperCard* stack. It handles initialization of variables and sets conditions that you wish to set up as you open the folder. In this case, a variable called blank (which holds 40 blank characters) is initialized, the menu bar is disabled, and a command is given to `go` to the first page of the folder:

"Autoexec" button:

```
var blank(40);
set blank = "
                    ";
nombar
go 1;
```

The last two buttons on the base page are "Go" buttons that aid in navigation—a left arrow and a right arrow. You create these by selecting "New" from the objects menu and choosing "Button" from the first pop-up menu.When the next pop-up appears, instead of selecting script as we have with the other buttons, select the first type on the list, "Go." You can then name these buttons "Previous" and "Next" if you choose, but what is really important is that you look through the icons presented in the next pop-up window until you find the appropriate arrow, and select it. On the next pop-up menu (the "Go..." menu) select "Previous" for the left pointing button, and "Next" for the right pointing one. This completes the base page.

The Data Page

We are now ready to create a data page like the one shown in Figure 6. The first step is to create page 1 of the folder by choosing "New" from the Page menu. This page will display all of the objects that were on the base page, but they are not accessible for editing except by choosing "Base Page" from the Go To menu. If you have followed all the instructions and keyed in all of the scripts properly, you won't have to go back to the base page again, but if you want to make changes, the "Base Page" menu option is always available.

There are 8 fields on each data page that are created in much the same way as those on the base page. The one difference is that each field must be named correctly for the scripts that we put on the base-page buttons to work properly. The names are: "numfld," "namefld," "yearfld," "typefld," "howmyfld," "condfld," "worthfld," and "totfld." Also the fields must be made large enough (i.e., be able to hold enough characters) for the data. For example, 4 characters would be plenty for the "numfld," but the "namefld" should be closer to 35 characters. Use the data shown in Figure 6 to guide you.

Hand-Held Scanning

A major plus of these applications is that we can actually scan baseball cards directly into the stacks, projects, and folders if we choose. Thus the software can become a more graphic record of the collection.

We made use of relatively inexpensive hand-held scanners from Logitech and ThunderWare, and found them to be generally satisfactory. Because these hand-held scanners work with resolutions in 100 dot per inch (dpi) and the screen resolution of the Macintosh is 72 dpi, scanning at actual size presented some minor obstacles. On the other hand, using a Macintosh II, *SuperCard*, and ThunderWare's LightningScan proved to be a great match up for high-quality gray-scale scans. Any gray-scale scanner (e.g., Apple's flat-bed) or color scanner (e.g., ProVis scanner using video camera for input) allows for improved performance with *SuperCard*. The scan on this issue's cover was done using the ProVis scanner.

HyperCard Scanning

We used both the Logitech ScanMan and ThunderWare LightningScan hand-held scanners as desk accessories because it is the easiest method for importing graphics to *HyperCard*. The plan is simply:

- Go to the scanner desk accessory from the "Baseball Card Collection" stack
- Scan the card
- Copy the image to the clipboard
- Return to *HyperCard* and paste it, place it, crop it, etc. with *HyperCard's* Paint tools

This plan only worked in a very limited way. To understand the difficulties you must understand the basic features of these hand-held scanners. First of all, they make four different resolutions available: 100, 200, 300, and 400 dpi. *HyperCard* only supports, without XCMD's and XFCN's, 72 dpi *MacPaint* style graphics. Even at the lowest resolution of 100 dpi, any scan from these scanners must be either compressed to 72 percent of the original size (thus losing some resolution) or expanded by 138 percent (thus making the image almost half again as large as the original). Both ScanMan and LightningScan ran into this problem. Increasing the resolution on the scanner to 200 dpi or greater did not help, it just made compression and expansion of the image a greater problem.

The solution we employed was to use the "Line Art" setting at 100 dpi—the worst quality the scanners could produce. The result is what you see in Figure 1.

It was interesting that ScanMan's "Fit in Window" and LightningScan's "Fit to Window" options

—continued on page 60

HyperLink Tips

A Single Control Button for Easy Radio Button Handling

Radio buttons are handy devices for Macintosh programmers and they are convenient for users. They provide a visual indication of which of several options is selected, make it easy to choose a different option, and make it impossible to choose more than one option. Most Mac users are already familiar with them, so no instructions are necessary. The mutually exclusive nature of radio buttons, however, is not built-in to *HyperCard* or *SuperCard*. Without special programming on the part of the scripter, the user can select two or more radio buttons at once. This violates Apple's user interface guidelines and can create confusion for the user.

Whenever a radio button is clicked, somehow the previous choice must be unselected. I have seen several methods for doing this, but all of them involve custom scripting for every application and are difficult for the novice scripter to use in his own stack. (Difficult for me, too.) While pondering this state of affairs, I suddenly came up with this trick. A few minutes of frantic scripting proved the concept, and now radio buttons no longer intimidate me.

The Trick

The trick is to use a large, transparent button (I call it the "Radio Control" button) over the top of a group of radio buttons to capture the user's click (see Figure 1). This single overlay button does the work of deselecting the last radio button clicked and selecting the new choice, so that the radio button scripts can do their thing without worrying about which other button might be selected.

How It Works

When the user clicks on the "'Radio Control" button, this is what happens:

- The click location is temporarily stored.
- The button hides itself.
- The script clicks at the last place the user clicked. This de-selects the previous choice.
- The script clicks at the new place (stored in step 1). This selects the new choice.
- The current click location is saved so step 3 will succeed next time around.

Editor's note: This issue's *HyperCard* and *SuperCard* tip comes from Doug Weathers of Tigard, Oregon, and is called the "Radio Control" button.

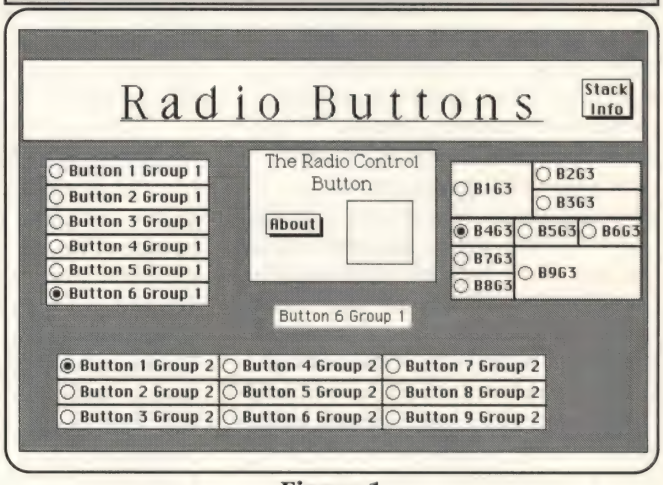


Figure 1

This screen has different "Radio Control" buttons on it with each one managing its own independent radio button group. No scripts are required either at the stack, background, or card level for this to work properly.

Listing 1 - Script for the "Radio Control Button"

```
on mouseUp
  put the clickLoc into thisClick
  hide me
  click at (last word of the ¬
  short name of me)
  click at thisClick
  get the short name of me
  put thisClick into last word of it
  set name of me to it
  show me
end mouseUp

on newButton
  get the short name of me
  put "0,0" into last word of it
  set name of me to it
end newButton
```

- The button shows itself so the user can click on it again.

The current click location is stored in the name of the button (the last word of the name, to be precise) so

the neophyte or lazy scripter doesn't need to mess with hidden fields or global variables. This does mean, however, that neither you or the user should mess with the name of the "Radio Control" button or it might get very confused.

To get this to work, the stack author does have to follow a few rules.

- The radio buttons must have their "Auto hilite" set to true. Otherwise, they will never light up.
- Every pixel underneath the "Radio Control" button must belong to a radio button. No empty space is allowed between or around the radio buttons.
- Every pixel inside a radio button must also be underneath the "Radio Control" button.
- When the "Radio Control" button is first placed over the radio button group, all radio buttons underneath *must* be unselected—i.e., their "hilite" property set to false. This is the default condition when a radio button is created, so it should be no problem. Also, "0,0" should be the last word of the "Radio Control" button's name. This is done automatically by an on `newButton` handler in the "Radio Control" button's script, so if you paste the button in to your stack and then re-size it over the radio button group, it is done for you (see Listing 1).

The easiest way to make sure these rules are followed is to create your radio buttons first. Make sure they are all unselected, then group them together into a *perfect* rectangle, with no space showing between them. (You can overlap them if you must, but this could get messy and I don't recommend it.) Now paste a copy of the "Radio Control" button and drag it over the top of your rectangle of radio buttons. Align the top-left corners of the "Radio Control" button and the rectangle, then drag the bottom-right corner of it until it is exactly aligned with the bottom-right corner of the radio button rectangle. Be sure the "style" of the "Radio Control" button is transparent, and its "Show Name" is false. Choose the Browse tool and click on your default choice.

Michel on SuperCard

Creating Floating Palettes in SuperCard

by Steve Michel

Welcome to the first installment of "Michel on *SuperCard*." Each issue, this column will take a look at a different feature of *SuperCard*. We'll stay away from areas where *SuperCard* and *HyperCard* work the same way, and concentrate instead on using features unique to *SuperCard*.

Working with Palettes in *SuperCard*

One of *SuperCard's* most exciting abilities is that it lets you create floating palettes. These palettes, similar to *HyperCard's* tool palettes, float above other windows, and let you create tools that can work in a variety of different situations. This lets you create tools in one place, and use them from other windows without having to copy buttons from card to card or install scripts in your "Home" stack.

Palettes are simply special types of *SuperCard* windows, and their behavior only differs from other types of windows in a couple of ways. Before going into palettes, let's take a look at the tools *SuperCard* gives you for working with windows in general.

- `This window` - This identifier always refers to the current—that is, the top—window, as does `this project`. Typing `The name of this window` into the Message box returns the name of the current window.
- `The number of windows` This always returns the number of windows that are in the current project (that is, the one containing the top window).
- `The number of current windows` - Returns the number of open windows.
- `The TopWindow` - This function returns an identifier of the window that is currently on top.
- `The CurrentWindow of number` - Just as buttons and fields in *HyperCard* exist on "layers," so do *SuperCard* windows. They are numbered from front to back. In Figure 1, the top window is `currentwindow(1)`, and the back window is `currentwindow(4)`. Clicking on `currentwindow(4)` will bring it to the front (see Figure 2), changing its number to 1, and adding 1 to the number of all the other windows. Note that this number is distinct from the window's number, which refers to its order in the project that contains it.
- `The SetWindow Command` - This command takes one argument, the identifier of an open window, and makes that window the current window. From within

Steve Michel is the author of Steve Michel's SuperCard Handbook *(due in bookstores by the time this column appears) and* HyperCard: The Complete Reference *both published by Osborne/McGraw-Hill. He has created several* HyperCard *and* SuperCard *products, including* Port Authority, *which is available from Heizer Software.*

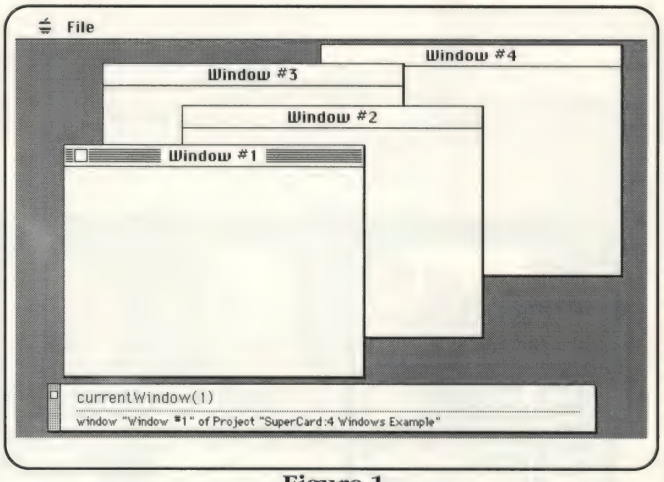


Figure 1

"Window #1" is the front-most window, and is therefore both window number 1, and currentWindow(1).

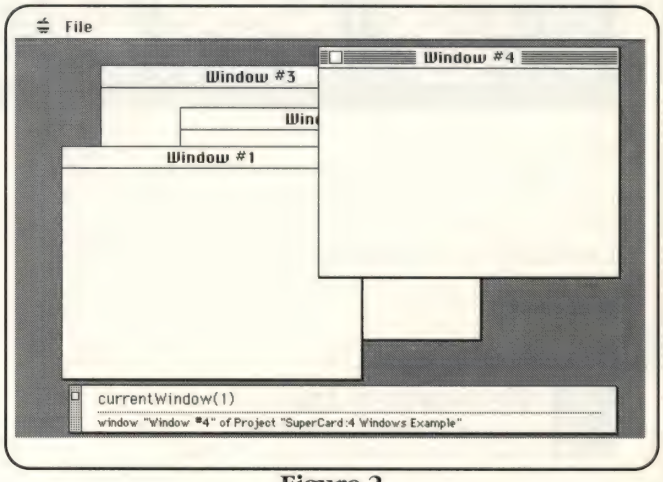


Figure 2

Even though it is "Window #4," because it is now the front-most window it is referred to as currentWindow(1).

Listing 1 - Script for the card containing the palette buttons

```
on mouseUp
  global theMode
  -- make sure a button was clicked
  if "button" is not in the long name of the target then
    exit mouseUp
  end if
  -- which button?
  get the short name of the target
  send "Go" && it && theMode to topwindow()
end mouseUp
```

Listing 2 - Script for the "Card" radio button

```
on mouseUp
  global theMode
  if the hilite of the target is false then
    put "card" into theMode
    set the hilite of the target to true
    set the hilite of cd button "Window" to false
  end if
end mouseUp
```

Listing 3 - Script for the "Window" radio button

```
on mouseUp
  global theMode
  if the hilite of the target is false then
    put "window" into theMode
    set the hilite of the target to true
    set the hilite of cd button "Card" to false
  end if
end mouseUp
```

a script it is roughly equivalent to clicking on a window to make it the front, or top window. After using this command to make a window the current window, things such as "this window," "this card," etc. all refer to that window, and messages are sent to the current card in that window.

So, how are palettes different from other windows? As I said earlier, they always float on top of the other windows, so that clicking on a window that is behind a palette does not obscure the palette. In SuperTalk, many of the tools discussed above do not work, or work differently with palettes.

The primary difference between a palette and any other kind of window is that a palette is *never the top window.* That is, if you have a window open with a floating palette on top of it, and type `topwindow()` into the Message box, the name of the other window you have open is returned, and not the name of the palette. The SuperTalk statement `this window` never refers to a floating palette, which is why the Runtime Editor's "Window Info" item doesn't give you info about a floating palette. Palettes also cannot be made the current window using the `SetWindow` command.

(Actually, the first part of the previous paragraph is not always true: the `topWindow` function and `this window` statement do refer to a floating palette if there are no other types of windows open.)

A Navigation Palette

In beta copies of *SuperCard*, Silicon Beach included a simple navigation palette that you could use to move through projects. They left that out of the release version, replacing its functions with several menu items on the Runtime Editor's Go menu. While the Go menu is handy enough, a palette can be very handy as well. You can copy it from project to project, freeing yourself from having to come up with different navigation buttons for each different project or window you create. You can also leave it on disk as a separate project, opening it when needed using the "Open Project..." menu item in the Runtime Editor.

Making a navigation palette is easy. You simply create a new window in *SuperEdit*, and select palette from the different window types in

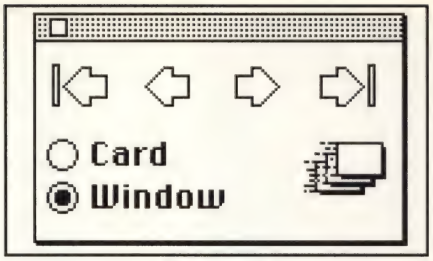


Figure 3

This is the floating navigation palette that you can create by following the instructions in this column. The four button ICONs across the top are SuperCard *versions of the standard* HyperCard *go first, previous, next, and last buttons. The "Card" and "Window" buttons are standard radio buttons. The "Scan" button ICON is not available in* SuperCard *and can be imported from* HyperCard.

the "Window Info" dialog box. Figure 3 shows such a palette. It has four standard navigation buttons, for going to the First, Previous, Next, and Last cards. Create these buttons on the palette, and give them those names. These buttons will have no scripts. Instead, type Listing 1 into the script of the card containing those buttons.

Next, create the two radio buttons as shown, naming one "Card" and the other "Window." Type the script in Listing 2 into the script of the button named "Card."

Type the script in Listing 3 into the script of the "Window" radio button. (This is the same script as that in the "Card" button, modified to handle the different button names).

We'll get to the "Show all Cards" button (in the lower-right corner in Figure 3) in a moment, but first let's take a look at how these buttons work together. The "Card" and "Window" radio buttons tell the other buttons how to work: if you have "Card" checked, the navigation buttons move you from card to card in the top window. If you have "Window" checked, then the navigation buttons let you explore the different windows in the current project. They do this by putting either "card" or "window" into the global variable `theMode`.

As mentioned earlier, the navigation buttons do not have their own handlers. Instead, the `mouseUp` handler in the card script takes care of things. This script uses the global variable `theMode`. It first makes sure

that a button was clicked, by checking the long name of the target. If a button was *not* clicked, the handler exits. The `mouseUp` handler simply sends the message `Go`, followed by the short name of the target (i.e., the navigation button you clicked), followed by either the word `Window` or `Card` to the top window.

For example, if you have the "card" radio button selected, and click on the "next" button, then the `mouseUp` handler sends `go next card` to the top window.

The last button on the palette, the "Show all cards" button is also simple to create. Its script is:

```
on mouseUp
  send "show all cards" to ¬
  topwindow()
end mouseUp
```

You may notice that the ICON used for the "Show all cards" button is not available in "stock" *SuperCard*. You can use the "Import Resource" command in *SuperEdit* to import all the ICON resources used in *HyperCard* into *SuperCard*. This command creates a new *SuperCard* project, and you then copy all the ICONs that were in *HyperCard* into your "SharedFile." This is a good thing to do if you plan on transfering many *HyperCard* stacks into *SuperCard*—otherwise, the converted stacks may look different.

You can create this window as part of another project, or all by itself in a project of its own. If you make it a project of its own—as is done on the disk—then all you have to do to open the palette is open that project. You can use the statement `Open Project "Go Palette"` or select "Open Project..." from the Runtime Editor's File menu.

Palettes and Tools

Another difference between palettes and other windows is in the behavior of tools. *SuperCard* features a number of different tools for creating buttons, fields, and graphics, as well as some tools for selecting those objects. In *SuperCard*, tools are always selected for one window only.

You can see this if you open a couple of windows. Type "`choose draw rect tool`" into the Message box. You have just selected the draw rectangle tool for the top window. Type "`the tool`" into the Message box, and *SuperCard* returns `draw rectangle tool`. Now click on a different window, and strike the Return key (because `the tool` should still be in line 1 of the Message box). It should return `browse tool`. Now choose the first window again, and hit Return again. *SuperCard* tells you that the tool for this window is the draw rectangle tool. This is pretty powerful stuff.

The difference between a palette and any other type of window is that the tool for a palette is virtually always the browse tool. This is what lets the Tools palette work from the Runtime Editor. When you choose a different tool from that window, only the tool of the `topwindow` (which, you'll remember, is only rarely a palette) is changed. If this were not true, you couldn't use a palette to change tools.

The one problem with this is that it makes it difficult to edit floating palettes in *SuperCard*. There's not a lot that can be done to make it easy to select different tools to use with floating palettes. Remember that clicking on a floating palette, even on its "title bar" makes that palette window 1. You can then change the tool of a palette window with a statement such as `set the tool of currentWindow(1) to pointer`, or whatever tool you need. Remember to use the statement `set the tool of currentWindow(1) to browse` when you want to test buttons. You can also use the `send` command to `send` commands to a palette. For example, if you want to cut a card from a floating palette, use the statement `send "cut card" to currentWindow(1).`

You can also make script editing easier in any circumstance by putting a handler into the script of your "SharedFile." This handler mimics *HyperCard 1.2's* "script peeking" features: if you hold down the Option and Command keys and click on something, you will open up the script of the object you click on. The handler that takes care of this is in Listing 4.

Listing 4 - Script for emulating *HyperCard's* script peeking

```
on mouseDown
  if the optionKey is down then
    if the commandKey is down then
      edit script of the target
      if "button" is in the name of the target then
        set the hilite of the target to false
      end if
      exit to SuperCard
    end if
  end if
end mouseDown
```

You can type this handler into the script of your "SharedFile" by typing `edit script of project sharedFile()` into the Message box in *SuperCard*. In *SuperEdit*, simply open the "SharedFile" (it is usually in a folder called "SC Pouch"), and then open the script of that project. By placing the handler in the script of your "SharedFile," it is available to all your projects, and works everywhere.

The `mouseDown` message is used to make this script peeking feel more like *HyperCard's*. The logic of this is pretty straight forward. It uses the target function to determine what you clicked on, and edits that script. The `if` test to `set the hilite of the target to false` is due to a bug in *SuperCard* 1.0 that leaves the button's "hilite" property true after you've edited the script. Note, too, the `exit to SuperCard` line that is necessary to prevent *SuperCard* from sending the `mouseUp` message to the target when you've finished editing the script.

Conclusion

As I stated at the beginning of this column, *SuperCard's* floating palettes are one of its best features. You can save yourself a lot of time by using palettes for such things as navigation tools. It's much easier to create a floating palette that you can use anywhere than it is to copy buttons from stack to stack. So the next time you are developing some utilities, consider developing them as floating palettes.

Xpanding HyperCard

Feel Constrained by ICONs? Here's Two Externals That Allow for Greater Flexibility in Both the Size and the Highlighting of Button Pictures

by James Paul

In the last issue we demonstrated how an XCMD can be used to handle *HyperCard* buttons of any shape. Now we're going to take another look at rectangular buttons, and see how we might be able to spice them up a little. We'll show how two external routines can be used to change the way a button looks when it is highlighted.

Each *HyperCard* button has an "Auto Hilite" property that is normally turned off when the button is created. With "Auto Hilite" selected, the button is highlighted when it is clicked on. I've often wished that this feature was turned on by default, because I believe in an immediate response for each action that a user takes. Lately I've seen several scripts and XCMD's that make button highlighting more exciting, especially buttons that have ICONs attached to them. Most of these XCMD's and scripts let us change how the highlighted version of the button looks, usually by using another ICON. I like this a lot, because most of my buttons have ICONs. But what about buttons that don't use ICONs or are too big for ICONs? We'll use an XFCN called `PushButton`.

The PushButton XFCN

What we want to do is change the way a button looks when it is clicked on, then restore the original appearance when the mouse button is released. This is what the `PushButton` XFCN does. We call it from a `mouseDown` handler (not `mouseUp`) in our button script. `PushButton` handles the highlighted appearance of our button when it is clicked on.

Here's what happens inside `PushButton`: First, the original (unhighlighted) appearance of the button is stored. Next, the highlighted version of the button is drawn over the original. Now the mouse position is tracked until the mouse button is released. If the mouse cursor moves outside the button rectangle, the button is unhighlighted by drawing the original appearance that was stored at the beginning. Accordingly, if the mouse moves back into the button rectangle, the highlighted version is drawn again. When the mouse button is released, the unhighlighted version is drawn, and `PushButton` returns a value to let us know if the mouse was released inside or outside of our button. The following is an example script of a button using `PushButton`.

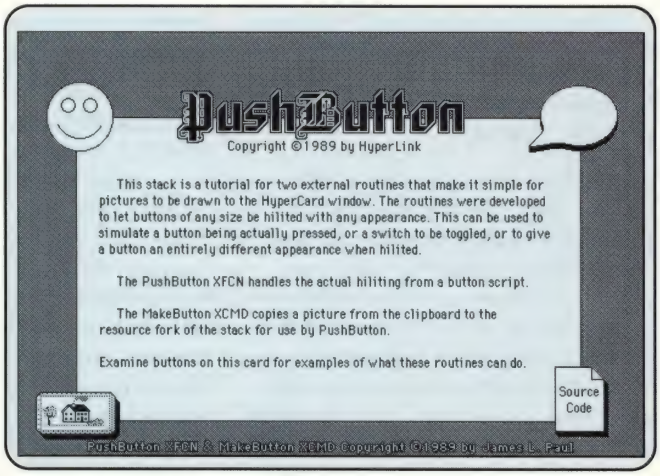


Figure 1
The main screen of the stack that features the PushButton and MakeButton XCMD's.

```
on mouseDown
  if PushButton("myButton") is "in" then
    play boing
  end if
end mouseDown
```

That's all `PushButton` does. It just decides whether the button should be highlighted or not, and draws what we tell it. So, how do we tell it what to draw? Well, we have an XCMD called `MakeButton` to let us do that.

The MakeButton XCMD

Making a highlighted picture of a button is pretty simple. Because we're not using ICONs, our button must be drawn with the paint tools and covered with transparent buttons. We need to draw two versions of our button, one for the normal appearance, and one

James Paul is chief programmer for Paul Software Engineering, and he is the author of Icon Factory *from HyperPress. He is also the author of numerous XCMD's and XFCN's, which include DoList, ListRes, and SortIt.*

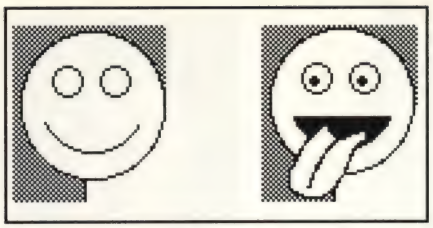


Figure 2

The picture on the left is drawn on the card level of the card shown in Figure 1. The picture on the right was also created with the HyperCard paint tools, but was cut to the clipboard where it was made into a PICT resource by the MakeButton XCMD with a script much like that shown in Listing 1. Then a script much like the one shown in Listing 2 is placed in a transparent button over the picture on the left, and each time the mouse clicked and held down over the button, the picture on the right appears in place of the one on the left.

highlighted. After we have done this, we copy the highlighted version to the clipboard, and call the `MakeButton` XCMD. `MakeButton` will store the picture of our highlighted button as a PICT resource in our stack. Once this is done, the `PushButton` XFCN can use the picture when it needs to draw the highlighted button. We only need to use `MakeButton` to store the highlighted appearance of our button, because the unhighlighted appearance is stored on the card.

Here's how `MakeButton` works: First, create the picture for the highlighted appearance of the button, and copy it to the clipboard. Then call the `MakeButton` XCMD. At this point the `MakeButton` copies the picture on the clipboard and determines whether a picture resource with the same name is already stored in the stack. If one is, it is removed so that we don't have more than one picture with the same name. Next, it adds the picture to the resource fork of the stack. Now the picture is stored for use by the `PushButton` XFCN. Listing 1 shows a script for a button that you can make to invoke the `MakeButton` XCMD.

So `MakeButton` just stores a picture of the highlighted version of our button in the stack as a PICT resource, so that `PushButton` can draw it. Is it really as simple as this? Well, yes and no. When a picture is made into a PICT resource, the rectangle of the picture is stored along with the picture itself. This means that the picture will normally be drawn at the same location from which it was copied.

Listing 1 - An example of a script using the MakeButton XCMD

```
on mouseUp
  ask "Name for highlighted button picture?"
  put it into pictName
  if pictName is not empty then
    MakeButton pictName
  end if
end mouseUp
```

Listing 2 - Using the PushButton XFCN with 3 parameters

```
on mouseDown
  put item 1 of rect of me into horiz
  put item 2 of rect of me into vert
  if PushButton("myButton",horiz,vert) is "in" then
    play boing
  end if
end mouseDown
```

What if we move the button to another place on the card? Or what if we draw and copy the button at a location different from where we want to use it? In these cases, `MakeButton` works the same way, and still stores the rectangle from which the picture was copied. The `PushButton` XFCN, however, has a way to take care of this problem. `PushButton` can take either one or three parameters. If the picture doesn't need to be drawn at a place different from where it was copied to the clipboard, it only needs the name for the PICT as its single parameter.

If, however, our button is at a different location from where we drew its highlighted picture, we need to tell `PushButton` where *that* location is. We do this by adding two more parameters, the horizontal and vertical positions of the upper-left corner of the button. Actually, it's a good idea to use these parameters even if you are copying the picture of the highlighted button from the same place where it will be drawn. This will ensure that the highlighted picture will always be drawn in the same place, even if it is moved later. Listing 2 shows an example of using all three parameters.

A Little Tech Talk

For those of you who are interested in learning how to write external routines, I thought I'd include a more technical description of how `PushButton` and `MakeButton` operate.

PushButton XFCN

- Check the number of parameters to be sure it's either 1 or 3.
- Use ZeroToPas on parameter 1 to get the picture name.
- Use GetNamedResource to get the PICT resource itself.
- Use DetachResource so the PICT is no longer in the resource map.
- Check ResError to be sure there were no errors.
- Get the GrafPtr to the *HyperCard* window.
- Get the rectangle of the PICT.
- Check to see if there were 3 parameters. If there were, then use ZeroToPas and StringToNum to get the horizontal and vertical parameters.
- If there were 3 parameters, adjust the PICT rectangle to the proper location.
- Calculate the proper values for a BitMap the same size as the PICT, and use NewPtr to allocate memory for the BitMap baseAddr.
- Use CopyBits to copy the unhighlighted button from the *HyperCard* window to our BitMap.
- Use DrawPicture to draw the PICT to the *HyperCard* window.
- Set a boolean "hilited" flag to true.
- Loop until the mouse button is released using the Button function. While looping, Use GetMouse and PtInRect to track the mouse position. Use DrawPicture or CopyBits to draw the appropriate picture, and keep the boolean

"hilite" flag current.
- When the loop is done, use CopyBits to restore the original picture to the *HyperCard* window.
- Use DisposPtr to release memory needed by the BitMap.
- Use DisposHandle to release the PICT.
- Use PasToZero in setting the returnValue to "in" or "out" depending on the value of the boolean "hilite" flag.
- Exit gracefully. (Wasn't that fun?)

MakeButton XCMD

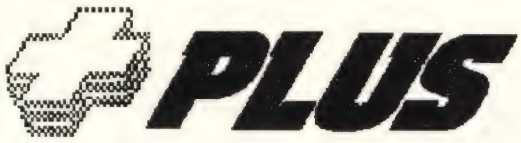
- Use NewHandle to create an empty handle for scrapData.
- Use GetScrap to get the PICT from the clipboard.
- Check the scrapLength and paramCount to be sure they are OK.
- Use ZeroToPas to get the pictName.
- Use Get1NamedResource to see if PICT already exists.
- If PICT does exist, Use RmveResource to remove it, then UpdateResFile(CurResFile) to flush it. Finally, use DisposHandle to release memory.
- Use AddResource and UniqueID to add the PICT as a resource.
- Check ResError to be sure there are no problems.
- Use WriteResource to write save the PICT.
- Use DetachResource on the PICT so we can throw it away.
- Use DisposHandle to release the PICT.
- Exit.

That just about wraps it up for this issue. The source code for the `PushButton` XFCN and the `MakeButton` XCMD are on the *StackProjects* disk for this issue, along with a demonstration stack. Please send any comments, and keep those ideas coming.

Here's how to contact me:

- c/o *HyperLink Magazine*, P.O. Box 7723, Eugene, OR 97401
- Compuserve 72767,3436
- GEnie J.PAUL

Introducing a Colorful Macintosh Object-Oriented Interface



by David G. Brader

Editor's Note: A copy of Runtime *PLUS* with some sample stacks is included on this month's *HyperLink StackProjects* disk for Macintosh.

What is *PLUS?* Olduvai Corporation calls *PLUS* "The Professional *HyperCard*-Compatible Multimedia Software ToolKit."

Professional? Don't let it scare you, working in the *PLUS* environment is very much like working in *HyperCard.* Professional does not mean hard to use. (Both the *PLUS* and *HyperCard* environments are quite different from working in the *SuperCard* environment which is gaining favor with many programmers).

HyperCard-Compatible? Generally, this means stacks built in *HyperCard* can be converted to *PLUS* stacks and run under *PLUS* without modification. This is true only for HyperTalk scripts that meet the common syntax rules for both environments—this is also true with *SuperCard's* acceptance of HyperTalk scripts.

MultiMedia? *PLUS* has all that *HyperCard* has (sound, graphics, and text) with the addition of color, high-resolution graphics, and full page printing.

A Macintosh Plus, SE, or Mac II with a minimum of one megabyte (MB) of random access memory (RAM) is required to use *PLUS*, but to run anything significant, 2.5 MB and a hard drive are needed. Let's jump right in and start looking at some of the tools in this ToolKit.

Sizing Up A Card

The size of a card can be set to any Mac screen size or to the printer paper size. So, setting up a normal letter or letter size document on screen is easy. If the card is sized larger than the *PLUS* window, scroll bars appear to allow navigation over the card's surface.

This variable size card allows the most exciting difference in the printing capability from *HyperCard*—the ability to easily print full pages.

Seven Window Types, One at a Time

The window type is controlled by the *PLUS* Programming Language (PPL) script. Generally, the window type is set in the stack with an `on openStack` script. This is a good spot to set other *PLUS* parameters, such as the palette and window rectangle size, as well. The seven window types include standard, zoom, dialog, plain, shadowed, nosize (no resize block in lower right-corner of the window), and rounded (with a variable to specify how round). As *HyperCard, PLUS 1.1* can have only one stack window open at a time.

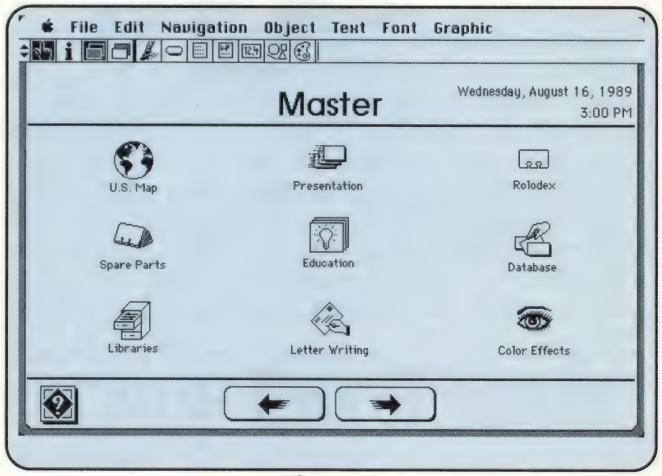


Figure 1

This screen picture shows the "Home" stack's first card as supplied with PLUS version 1.1. The icons shown activate sample stacks that come with PLUS 1.1. Some of them require a Mac II and color monitor to run.

On Toolbox Row

The tool box is in the form of a horizontal row of icons to the left followed by the message box to the right. This toolbox/message box row can be moved vertically on the screen with a handle on the left side. When in browse mode (see Figure 1), the icons shown left-to-right are browse mode, get info of selected object, card level selector, background level selector, paint tool, button tool, field tool, WPfield tool, DBfield tool, Draw object tool, Paint object tool, and the remaining area to the right is the message box. Whenever one

David G. Brader has over twenty years experience with computers ranging from mainframes to micros, hardware design, software systems and applications design and serves as HyperLink's *Publisher.*

of the seven tools is active with a corresponding object selected, the toolbox bar changes to show the icons for the subtools/options of the selected tool. This may sound hard to grasp, but it is quickly understood with mouse in hand.

Paint Objects

Paint Objects are containers for bit-mapped graphics. Imported graphics can be modified in a Paint Object (unlike Draw Objects). These Paint Objects can be designed as irregular shaped buttons. A shortcoing, however, is that the transparent pixel locations of a Paint Object do not respond to the mouse.

It would be nice if you could import or draw a Paint Object to "sensitize" the non-transparent pixels, and then select an option that would not show the paint object, but would still respond to the non-transparent pixel locations. This would allow irregular buttons to be placed over high-resolution graphics on a *PLUS* stack's card (or over an imported high resolution Draw Object graphic). This would be useful for Macs with high resolution screens and/or color.

For 72 dot per inch (dpi) monochrome Mac screens, irregular graphic button areas can be created from imported graphics using a paint object. The technique involves a three step process. First, size and position the paint object rectangle and import the graphic that will be the irregular button. Second, use the graphic as a guide to black out the areas that are to be sensitized (the active irregular shaped button) with the paint tools. Third, import the same graphic again. The sensitized areas remain, but the graphic reappears without the blacked out areas. Now, the irregular shape respond to `mouseUp`, `mouseDown`, etc.

Draw Objects

Draw Objects are containers for graphics imported as PICT, PICT2, EPSF (including color) files. (By the way, these file types can be imported and pasted on the card or background layer too.) A Draw Object's entire rectangle area is active when used as a button. A Draw Object's graphic cannot be modified. The *PLUS* painting capabilities, Draw Objects, and Paint Objects should prove exciting to those that are graphically oriented.

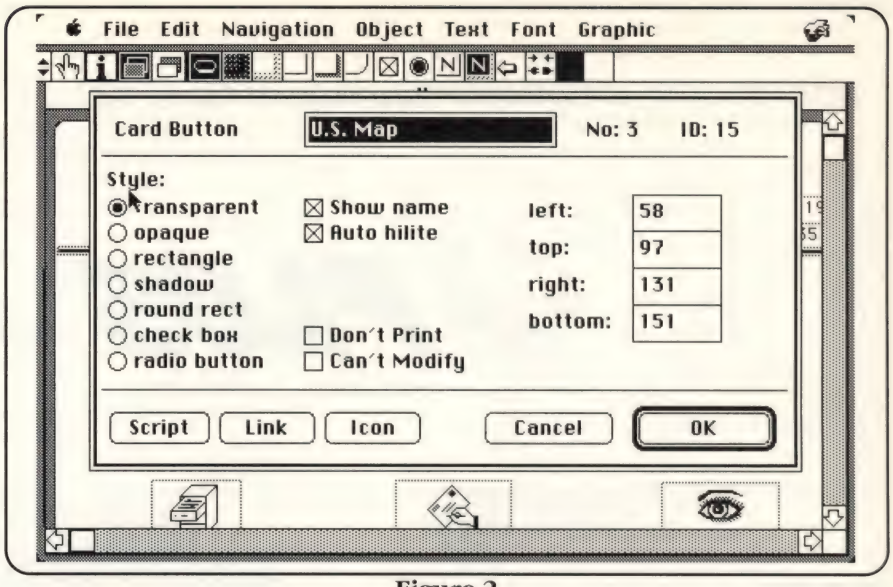


Figure 2

Here is an open object dialog box. All button, field, paint, and draw objects include the rectangle coordinates.

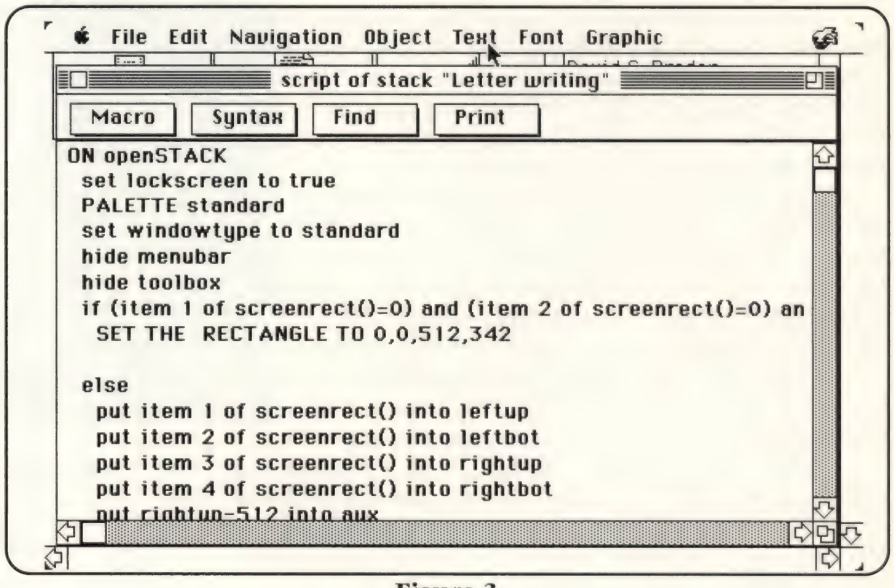


Figure 3

The "Script Editor" window is shown with the PLUS *home stack script in a bold font. In this script, you can see the stack's window parameters being set.*

WPfields and Others

The word processing fields (WPfields) allow mixed fonts and styles rather than just one selected font and style per regular text field. Standard fields are also supported.

Don't miss understand *PLUS* Database field objects (DBfields). They do not give *PLUS* any more database structure than *HyperCard*. DBfields are just like standard fields with the addition of data validation and/or format correction of any information placed in the fields.

A DBfield can be set to accept a certain number of any characters (alphanumeric) or a certain pattern of numeric characters including a decimal point and optional sign character. The numeric data checking looks most useful.

If a DBfield is setup for a specific date format and a date is entered in a different but recognized date format, the field will convert the date to the selected format. If the information entered into this field is not a recognized date format, the field shows an erroneous date.

Object Info Dialogs

The information box for objects appearing on cards has a few new features for those of you used to *HyperCard* (see Figure 2). Most important is that the size of the object can be specified with top, left, bottom, and right pixel location values. These are also new "Don't Print" and "Can't Modify" check boxes. Other than these additions, it feels just like home.

Script Editor

The script editor window in *PLUS* has several nice features that set it apart from *HyperCard*. Font size, style, and attributes can be changed. While the script editor window is active, other menu tools can be used. As an example, script text previously stored in the scrapbook can be accessed, copied, and finally pasted in the script editor window while it remains open.

The top of the window has four buttons which activate pulldown menus. The Macro pulldown will copy, at the current cursor location, standard script portions like `repeat with`, followed by a blank

line, followed by `end repeat`. The cursor is left after the `repeat with` so you can fill in the blanks. There are macros for all standard handlers, if-then-else structures, and repeat loops. The Syntax pulldown menu lists all PPL commands, functions, operators, and properties. When one of these is selected it is inserted before the current location of the cursor in the script editor window. The Find and Print pulldown menus contain no surprises.

Objects of Color

On the appropriate Macintosh systems, *PLUS* supports 256 color, 16 color, 16 grey levels, and black and white. WPfield text can be selectively colored as well as the foregrounds and backgrounds of most other objects.

Importing *HyperCard* Stacks

As one would expect, most stacks require some modification to work with *PLUS*. In most cases where we tried to convert a functioning *HyperCard* stack to a *PLUS* stack there was failure. These same stacks worked fine on the same machine in *HyperCard*. It would seem some of the differences between PPL and HyperTalk cause the system to crash with a converted stack on any Mac. The problems in this area appear to be surmountable.

Development Stress

During our testing of stacks under development in *PLUS*, the system would freeze—you could move the cursor around, but the mouse button had no effect nor did the keyboard. Apparently, in some cases, stack construction errors cause this behavior, rather than causing a recoverable error message

. We had one instance, when we placed a Paint Object with a new script over a Draw Object on the "Home" card. When we went to browsing mode *PLUS* quit with an unexpected error. Trying to restart *PLUS* from the Finder caused a system error. Restarting the system caused a system error because the *PLUS* "Home" stack was set as the startup file.

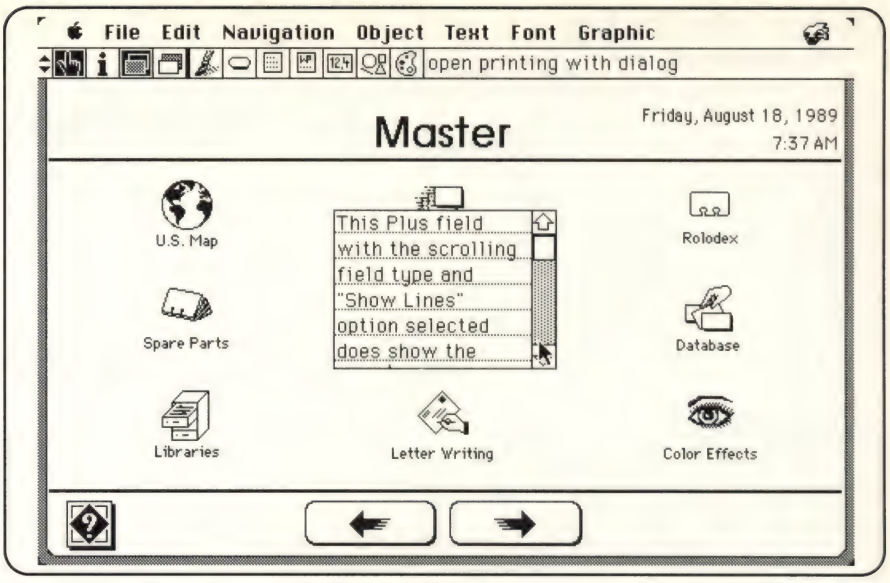


Figure 4

The scrolling attribute and show lines attribute for a field work together in PLUS—*unlike the current version of* HyperCard.

Finally, we restarted from a different disk, replaced the *PLUS* "Home" stack with the original copy and restarted. All changes we had made to the "Home" stack were lost

For those who have gotten used to the way *HyperCard* takes care of us, this sort of experience is disconcerting to say the least. The present *PLUS* environment is reminiscent of programming in assembly language or FORTRAN environments. From this aspect, during development, *PLUS* is a minus with respect to *HyperCard*. In all fairness, however, well constructed stacks seem to run just fine in *PLUS*.

Unfortunately, the "Help" stack supplied with *PLUS* has a problem. Mac users automatically use the Command key combined with the "F" key to activate the find function. Doing this in the "Help" stack crashes the system. The final proof as to a software product's viability and its final acceptance comes in its handling of unexpected events. And, in the Macintosh arena, this acceptance is inextricably tied to its following of the established user guidelines.

PLUS is young and full of potential. For those wanting the extra features not found in *HyperCard*, it is worth putting up with the current environment—given that *PLUS* continues to evolve. We are sure that feedback from us and other users of version 1.1 will result in fixes to the *PLUS* environment.

A Runtime *PLUS*

Stack developers do not need to worry about potential customers that do not own a copy of *PLUS*. The Runtime *PLUS* program is a version of *PLUS* that has been locked permanently into user level 2 (typing). The tools in the toolbox row have been deactivated as well. This version of *PLUS* can be distributed free with your stacks. The Runtime *PLUS* program, when present, starts automatically when a *PLUS* stack is opened from the desktop.

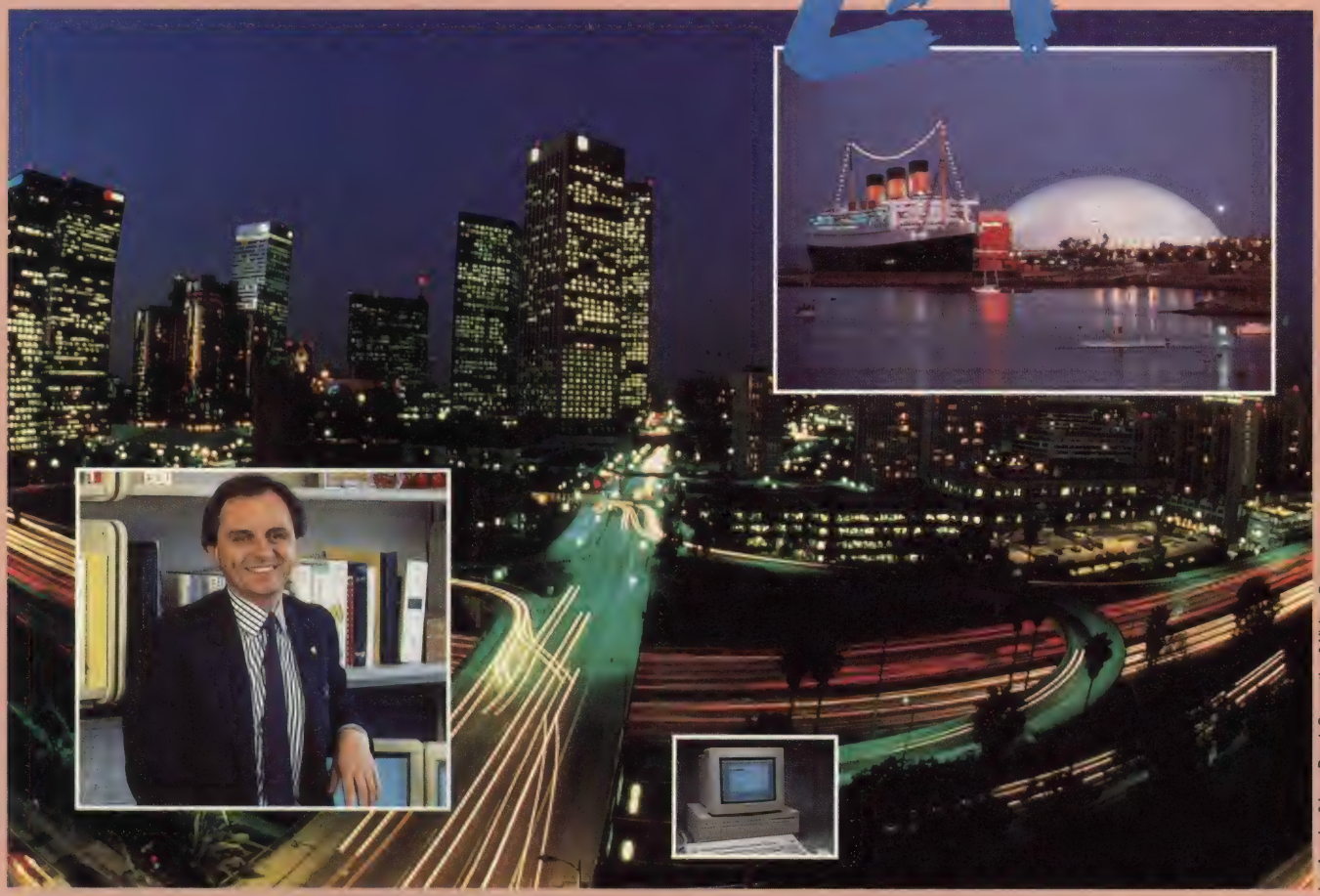
Documentation

Basically, the two manuals that come with *PLUS* are well organized and easy to use. Unfortunately, there are a few spots where the correct information got lost in the translation. Most of the errors are only bothersome to a magazine editor or English professor and are easily understood–poor editing. A few errors however may confuse new users–poor technical editing.

All in All, It's a *PLUS*

HyperLink will extend its coverage to include *PLUS*. Starting next issue, our *StackProject* stacks will be tested for compatibility. The ones that will work will be noted. When the new IBM version of *PLUS* from Format Software is introduced, we will tell you all about it right here in these pages.

Macintosh® LA



Los Angeles & Long Beach Convention & Visitors Bureaus

For Three Days This Fall, the Power Is Yours.

Get ready, L.A. Mark it down, Orange County.

MACINTOSH BUSINESS CONFERENCE & EXPOSITION

November 29-December 1, 1989
Long Beach Convention Center
Long Beach, California

From November 29 through December 1, the Macintosh universe is coming to you.

Come and be a part of the only national Macintosh conference and expo specifically designed for corporate and professional users. You'll learn exactly what you need to know to put your Macs to work most effectively. And you'll be able to test the latest hardware and software products from Apple and more than 100 leading vendors in the Macintosh market.

Don't miss MBC&E. It's your best chance to discover how to make the power of Macintosh work for you.

Call 1-800-262-3378* today for tickets or information.
*(617-860-7100 in Mass.)

Baseball Card Collection

—continued from page 45

were different. ScanMan's "Fit in Window" meant that the graphic would be expanded by 138 percent of the original size—i.e., each dot on the 72 dpi screen would be a mapping of a dot from the 100 dpi scan. LightningScan's "Fit to Window" option, however, turned out to be one that created an actual-size scan, and their default was the 138 percent enlargement.

This difference in nomenclatures aside, the scanners' operation was comparable—in fact the two scanners (one is shown on the cover of this issue) are even virtually identical in appearance. The only hardware difference of note is that the ScanMan uses a red correction system and LightningScan a green one. This means that red items on the original do not scan well using ScanMan. Look at the color baseball card on the cover of this magazine—the red words "hank aaron" and "Milwaukee Braves" do not show up when scanned with the Logitech scanner.

There is also a caveat that must be pointed out in Logitech's ScanMan—when you are running MultiFinder and select the desk Accessory you *must* hold down the Option key as you select it or it does not recognize the available memory, and does not work. This information is not documented but was supplied by Logitech technical support. It really ought to be in the manual or at least in a "Read Me" file, but it wasn't.

SuperCard Scanning

SuperCard supports many more types of graphics than *HyperCard,* and thus is much more flexible in terms of scanning options. It can import—directly with *SuperEdit—MacPaint,* Tagged Image File Format (TIFF), and PICT files. Both the Logitech ScanMan and ThunderWare LightningScan will save files in all three of these formats, so importing can be handled from *SuperEdit.* To get you started, we've included a version of the "Baseball Card Collection" on disk that uses a semi-automatic import procedure to import color and gray-scale images into the project. This version requires a Mac II with enough memory to run *SuperCard* in a 2 megabyte MultiFinder partition. Because of the limited number of Mac II's with *SuperCard* out there, we have not tried to include these routines in the pages of the magazine. We do, however, offer the *SuperCard* project on disk for anyone who is interested (see the inside back cover for information).

In the version of the "Baseball Card Collection" presented here, *SuperEdit* automatically installs the "Runtime Editor" when you convert the stack, so *SuperCard's* Paint palette is available. Thus, the approach outlined for *HyperCard* works fine for importing *MacPaint*-style graphics. Just use the desk accessories from within the *SuperCard* project, and paste and position the graphics.

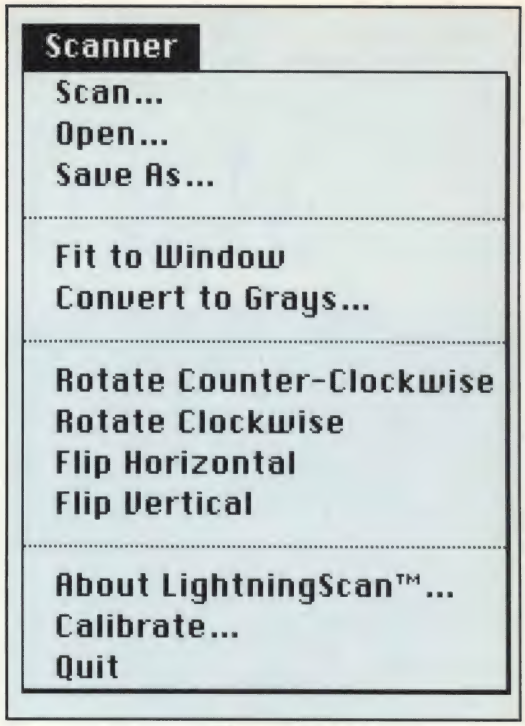


Figure 7

This is the ThunderWare LightningScan desk accessory's menu.

LinkWay Scanning

We only tested the IBM Logitech ScanMan hand-held scanner because ThunderWare's product is only for the Macintosh. ScanMan includes a product called *Paint Show Plus* with the scanner, which is a rather full featured paint program. As explained in some detail in the "Employee Records" article this software, along with *LinkWay's* own *LWCapture,* makes scanning a relatively easy process—but one that cannot be done from *LinkWay.*

The scanner can be fully accessed from with the *Paint Show Plus* software and you can even do your touch-up on the picture with their rather extensive paint tools. We found this to be a very serviceable paint program and recommend it, along with the scanner, as a nice adjunct to *LinkWay.*

For a detailed description of the entire process of scanning and importing the picture into your *LinkWay* folder, see page 24 of the "Employee Records" article in this issue. Except for the size of the "Picture" object, the procedures described are identical for this application.

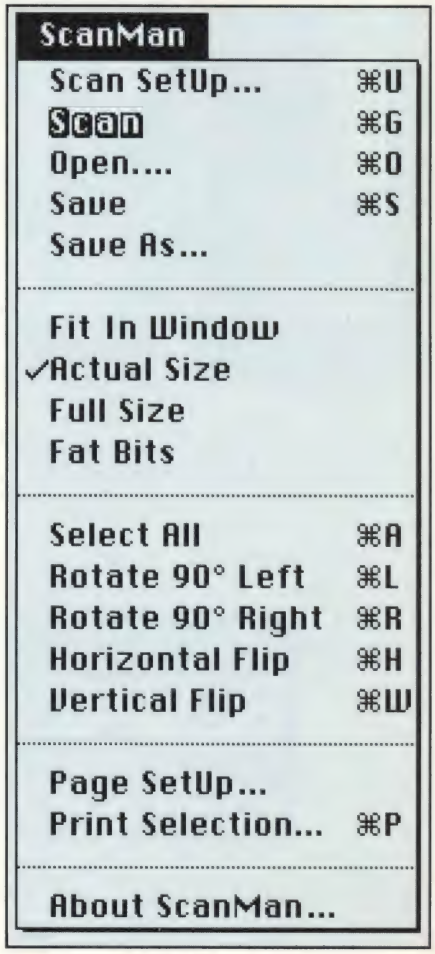


Figure 8

This is the Logitech ScanMan desk accessory's menu.

The *EASY* Way to Get Started with XCMDs:

Wild Things™ has all the examples, tools and templates you need.

40 Ready-to-Paste XCMDs

Animation: RecordPath, RecordLine, RecordGravity, DrawOval, RecordOval, DrawCurve, DrawSphere, RecordSphere, RecordCurve, Scene, Collision, Auto, RAuto, ShowData, BoundsCheck

Math: SinD, CosD, TanD, ASinD, ACosD, ATanD, ASin, ACos, DecToHex, HexToDec, Power, DegToRad, RadToDeg, Map3D

Statistics: Mean, Median, Variance, Deviation, Factorial, Probability

User Interface: UpCase, LowCase, PopUpMenu, Hello, WhichMac

Complete Source Code

Each XCMD is provided in FORTRAN, Pascal and C. Wild Things works with Language Systems FORTRAN™, TML Pascal II™, Think's Lightspeed Pascal™ 2.0, MPW™ Pascal 2.0.2 and 3.0, MPW™ C 3.0, Think's Lightspeed C™ 3.0, RMaker and Rez, MultiFinder™, and all Macintoshes Plus and above.

Powerful Tools

Four disks include WildIcons™ (a stand-alone icon editor to create, print, edit and move icons), ResCopy (a resource copier to move XCMDs, display icons and play sounds) and the current version of HyperCard®.

Time-Saving Templates

Wild Things automates the process of creating your own XCMDs and XFCNs, removing stumbling blocks from your path.

Nothing Hidden

Four interactive stacks demonstrate the use of all the XCMDs, and the source code is fully documented. See how to extract data from HyperCard fields, receive interactive information from the user and track the mouse. Utility routines simplify callbacks to HyperCard and illustrate advanced ways to speed up your stacks. They even take care of handles and pointers for you.

Wild Things is available from your local software dealer or direct from Language Systems.

LANGUAGE SYSTEMS CORPORATION
441 Carlisle Drive, Herndon, VA 22070
(703) 478-0181 fax (703) 689-9593

Guide to *HyperLink Magazine's* Available Back Issues

Each companion *StackSolutions/StackProjects* disk contains all the software described in the issue and more!

Volume 1, Number 2

June/July1988

Short Stacks
In praise of HyperCard's easy access to new users
—Joan Donaldson

Shafer On Scripting
Using script abstracting to increase efficiency and handling radio buttons with minimum scripting
—Dan Shafer

Buttons & Bows
A button to help you find the mouse location
—Mark Richardson

In the Cards
The author of HyperCard's "Help Stack" discusses when to put multiple backgrounds into your stacks
—Carol Kaebler

HyperCard Tips
HyperLink's idea swap meet
—William K. Balthrop

Xpanding HyperCard
Building a Sound Library with the ListRes XCMD
—James Paul

Hyper-Neuroanatomy
An educational stack that uses digitized brain sections
—Victor S. Johnston, PhD.

Class Stacks for Educators
A simple to use teacher's aid
—David G. Brader

A Different Approach to HyperTalk
An excerpt from his latest book—Danny Goodman's HyperCard Developer's Guide
—Danny Goodman

Help for HyperCard Printing
Featuring reviews of these printing utilities: Reports from Activision and HyperCONTROL from Nordic Software
—Roger Wood

Volume 1, Number 3

September/October 1988

Shafer On Scripting
Using script abstracting to increase efficiency and handling radio buttons with minimum scripting
—Dan Shafer

In The Cards
The author of HyperCard's "Help Stack" discusses text manipulation techniques
—Carol Kaebler

HyperCard Tips
A discussion of using "on idle" handlers
—Rbett Savage

Buttons & Bows
A button that proves to be part of a moving experience
—Kennetb S. Hulme

Xpanding HyperCard
Introducing SReplace, a new search and replace XCMD
—James Paul

Short Stacks
An easy to create letter-writing stack featuring push-button fonts
—Joan Donaldson

Budget Print A HyperTalk Utility
There is a tool for doing HyperCard reports that doesn't require extra expense: the HyperTalk scripting language
—Steve Roti

Venturing into Hypertext in The Lands of PC
An overview of Guide, Zoomracks, KnowledgePro, and our own HyperDocuments
—The HyperLink Staff

Adventure in HyperLand
Create your own custom adventure
—William K. Balthrop

HyperExpo and Product News
A report on the first ever HyperExpo and a look at some of the best stackware at the show
—Roger Wood

Volume 1, Number 4

November/December 1988

Short Stacks
Using HyperCard's built-in financial functions to make a Loan Calculator
—William K. Balthrop

Shafer On Scripting
Techniques for doing hypertext with HyperTalk
—Dan Shafer

In The Cards
A stack that helps Apple's office keep track of John Sculley
—Carol Kaebler

HyperCard Tips
Self-reflective HyperTalk or what scripts would M. C. Escher have made?
—Rbett Savage

Xpanding HyperCard
The Pad XFCN in a comparison of HyperTalk and Pascal
—James Paul

What ORACLE Foretells for the Macintosh
Big dividends for developers
—Dan Shafer

InventoryStack
Can your business keep track of its products? Now you can with your own InventoryStack
—William K. Balthrop

HyperCard and the 1-Megabyte Macintosh
Getting the most from a minimum configuration
—John DiBiasi

HyperCard and the Apple Scanner
How to integrate the Scanner with our InventoryStack
—David G. Brader

Reviews of HyperCard Products
A look at 101 Scripts & Buttons, HyperTools, ManorTools, HyperAnimator, and HyperCard Toolkits from APDA
—Roger Wood

Volume 2, Number 1

January/February 1989

Shafer On Scripting
Hypertext in HyperCard revisited; a look at an indexing technique
—Dan Shafer

HyperCard Tips
Your hot tips can earn you money and fame
—Kevin Altis
—James Baggott

Short Stacks
Use this "Personal Financial Statement" stack to calculate your net worth—it may be more than you thought
—William K. Balthrop

Xpanding HyperCard
The HyperCard Toolbox and the glue routines
—James Paul

HyperCard Haiku
The medium is the message
—Rbett Savage

Using ORACLE's Hyper*SQL
The nuts and bolts of a new Macintosh product
—Dan Shafer

Searching Stacks with Keyword in Context
Expanding HyperCard search capabilities
—Steve Roti

Networking with HyperCard—First Steps
If you think HyperCard and networking are not compatible, read this
—David G. Brader

HyperCard and CD ROM
A look at the present state of this exciting melding of technologies, featuring the BMUG PD-ROM and the Xiphias Time Table of History
—Roger Wood

Reviews of Products
This issue's products include ColorCard, Stack Cleaner, RDAide, and Tax Stacks

Volume 2, Number 2

March/April 1989

Shafer On Scripting
Moving with style among the objects
—Dan Shafer

HyperLink Tips
Updating a "Time" Field
—Rodney Magnuson

Interfacing the Future
The author of 101 Scripts and Buttons for HyperCard *begins a new series addressing the issues of interface design*
—Craig Ragland

HyperCard Haiku
Variable varities—speed and using globals
—Rbett Savage

Short Stacks
Planning your retirement can be as easy as filling in the blanks
—William K. Balthrop

Xpanding HyperCard
An in-depth look at an ICON animating XCMD from the author of Icon Factory
—James Paul

SQL Basics, Part 1
Learning about Structured Query Language (SQL) can help you link to databases outside the Macintosh domain
—Dan Shafer

The Big Message Box
Sometimes Hypercard's single-line message box binders as much as it helps—this is what HyperCard needs
—Kevin Altis

Creating a Selective Mailing List
Here's help from a binary search in Hypertalk
—Larry Walker

SuperCard—A Special Report
A peek inside Silicon Beach's new HyperCard "kissin' cousin"
—Roger Wood

Reviews of Products
Reviews of the Manhole, Hypertext for Beginners, *Focal Point II, Organizer+, Client, and CONTACTS!*

Volume 2, Number 3

May/June 1989

Shafer On Scripting
Overriding HyperCard's message-passing scheme
—Dan Shafer

InterfacingtheFuture
Creating easy to use control devices in HyperCard
—Craig Ragland

Short Stacks
"Securities Grapher" is a stack that makes practical use of the Paint tools
—William K. Balthrop

Xpanding HyperCard
A look at two commercial products with HyperCard expanded horizons
—James Paul

SQL Basics, Part 2
Here are some how-to tips on using frequently encountered SQL commands
—Dan Shafer

The "Letter Learner" Stack
An introduction to neural networks
—Robert Orenstein

The "Fixed Assets" Stack
This depreciation calculator can save you time and money
—William K. Balthrop

A Look at IBM LinkWay
Speculation about a HyperCard workalike from IBM is finally answered...But how does it stack up?—Larry Burtness

Proviz and HyperScan
Scanning is a snap with Proviz, HyperScan, and your video camera
—David Brader

Culture 1.0
What is a musical palindrome? What medieval scourge killed Petrarch's sweetheart, Laura? Find out in Culture 1.0

CompileIt!
How does the first HyperTalk compiler meet the needs of scripters?

Volume 2, Number 4

May/June 1989

Shafer On Scripting
How do I call thee? And when thou changest?
—Dan Shafer

HyperLink Tips
Shortcuts for changing HyperCard objects' levels and tips on using the HyperCard "Phone" stack to dial your modem
—Paul Chapman & HLM Staff

Interfacing the Future
Creating animation in HyperCard and SuperCard—Part 1 of 2
—Craig Ragland

Short Stacks
Qualifying for a loan can be stressful—see if you qualify before you go to your bank using the "Mortgage Qualifier"
—William K. Balthrop

Beginner's Card
Some plain talk about button properties for those just getting started with HyperCard
—Roger Wood

Xpanding HyperCard
Using an XCMD and an XFCN to bring non-rectangular shaped buttons to HyperCard
—James Paul

Videodisc Browser
Creating a videodisc controller with IBM LinkWay and HyperCard
—Larry Burtness

"ThingMaker" A SuperCard Project
Designing applications in SuperCard
—Chris van Hamersveld

Wild Things and Other XCMD's
A look at some recently released products that rely heavily on XCMD's and XFCN's

HyBase & HyperHit
These extensions speed up and shrink large HyperCard databases, but, alas, they require some study

HLM Product News
Short takes of commercial Personal Software Development products

StackProjects Disk Directory


Employee Records


Employee Records.SC


Baseball Card Collection


Baseball Cards.SC


HyperLink Tips V2,#5


SuperCard Custom Menus


Xpanding HyperCard V2,#5


Michel on SuperCard V2,#5


PLUS Runtime

Article	Page
Employee Records	11
SuperCard Custom Menus	15
Baseball Card Collection	39
HyperLink Tips V2,#5	46
Michel On SuperCard V2,#5	48
Xpanding HyperCard V2,#5	51
PLUS Runtime	55

HYPERLINK MAGAZINE

The Complete Resource —

HyperCard! SuperCard! IBM LinkWay!

Complete Applications • Tips • Tutorials • Reviews

Two Ways to Subscribe —
Magazine Only *Or* Magazine & Disk

HyperLink Magazine

P.O. Box 7723 • Eugene, OR 97401 • USA

NAME _____

COMPANY _____

ADDRESS _____

CITY _____

STATE + ZIP CODE _____

COUNTRY _____

TELEPHONE _____

Payment By: ☐ VISA ☐ MasterCard ☐ Check/Money Order
☐ Purchase Order Number _____

Credit Card # _____ Exp Code _____

Signature _____

Magazine ONLY —	**Magazine & DISK —**
☐ 1 Yr $25 ☐ 2 Yr $45 ☐ 3 Yr $60	☐ 1 Yr $73 ☐ 2 Yr $129 ☐ 3 Yr $177
HyperLink is a bi-monthly publication	HyperLink is a bi-monthly publication

(Please note, at the present time, disks are only available for the Macintosh computer)

Added Air Shipping Rates for International Orders
All orders must be in US funds.

Canada/Mexico, add per subscription year		**All Other Countries**, add per subscription year	
Magazine only	Magazine and Disk	Magazine only	Magazine and Disk
$15	$21	$36	$54
(Back Issues Each Mag. add $3.00 and Each Disk add $3.95)		(Back Issues Each Mag. add $6.00 and Each Disk add $3.95)	

Subscribe Now Call Toll Free!

1 (800) 544-0339

Use VISA or MasterCard

Yes! We have Back Issue Magazines & Disks

ALSO Please send the following back issue products —

HyperLink Magazine back issues: Add $4.95 for each magazine.

☐ Vol. 1 #2 ☐ Vol. 1 #3 ☐ Vol. 1 #4 ☐ Vol. 2 #1 ☐ Vol. 2 #2 ☐ Vol. 2 #3 ☐ Vol. 2 #4

StackSolutions/StackProjects back issue disks (Macintosh): Add $14.95 / disk.

☐ Vol. 1 #2 ☐ Vol. 1 #3 ☐ Vol. 1 #4 ☐ Vol. 2 #1 ☐ Vol. 2 #2 ☐ Vol. 2 #3
☐ Vol. 2 #4 ☐ Vol. 2 #5 (for this issue)

Total $ _____

Please mark areas of interest to you: ☐ HyperCard ☐ SuperCard ☐ IBM LinkWay

Please note if for any reason a refund is to be made on this subscription, the value of any premiums and any delivered issues are subtracted from the refund.

Office Use Only:
Auth# _____
Ref# _____
Total _____

Macintosh® In Business

NEW 256-Page Guidebook - Free

The MicroGroup offers a balanced line of high technology and general business products resulting from solving our own productivity and production problems where outside solutions were unavailable. Our all new Spring '89 Guidebook was produced on Macintosh - of course!

Macintosh® In Manufacturing

We believe office and factory automation are mandatory components of customer service. We also recognize the necessity of applying automation as the only solution to our continued growth and profits. The most important lesson we learned is that simplicity and ease-of-use are the only considerations in developing any automation system.

Patented 1stKEY™ Database Software

Regardless of your size, 1stKEY can produce just about anything that you will ever need in a business database system - FAST. 1stKEY can provide MIS the productivity of true relational workload distribution and easy to use reporting across platforms. Now anyone can quickly design systems that can easily interact with other systems - even remote sites. 1stKEY features powerful patented FUZZY Triage FIND™ for performing inexact weighted searches across four fields.

MacBOT™ Robotic Systems

The MacBOT-8 data acquisition and control system was created as a flexible tool to apply the power of Macintosh and HyperCard® to your everyday automation requirements - "Personal Automation." MacBOT robots and robotic components can be applied to push-pull and rotary motions and multiple axis devices that look like their larger counterparts. Complete weighing systems available.

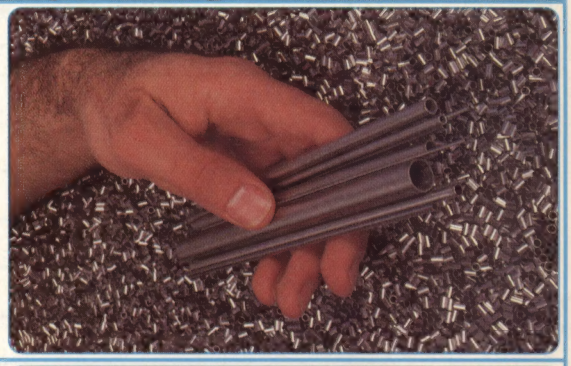
the MicroGroup® Products & Services Specifiers Guidebook

Abridged Pocket Version

Precision Metals

ALL-TUBE® Corporation

Stainless Steel Tubing
Luer and Tube Fittings
Precision Fabrication
Metalworking Tools



Office Automation

1stDESK® Systems, Inc.

Information Management Systems for the Macintosh®

Factory Automation

MacMOTION, Incorporated

Data Aquisition
Motion Control
Desktop Robots

7 Industrial Park Road Medway, MA 02053 **1-800-255-8823**

Our 256 page Guidebook is also in the >>>THOMCAT<<< Thomas Register Catalog File